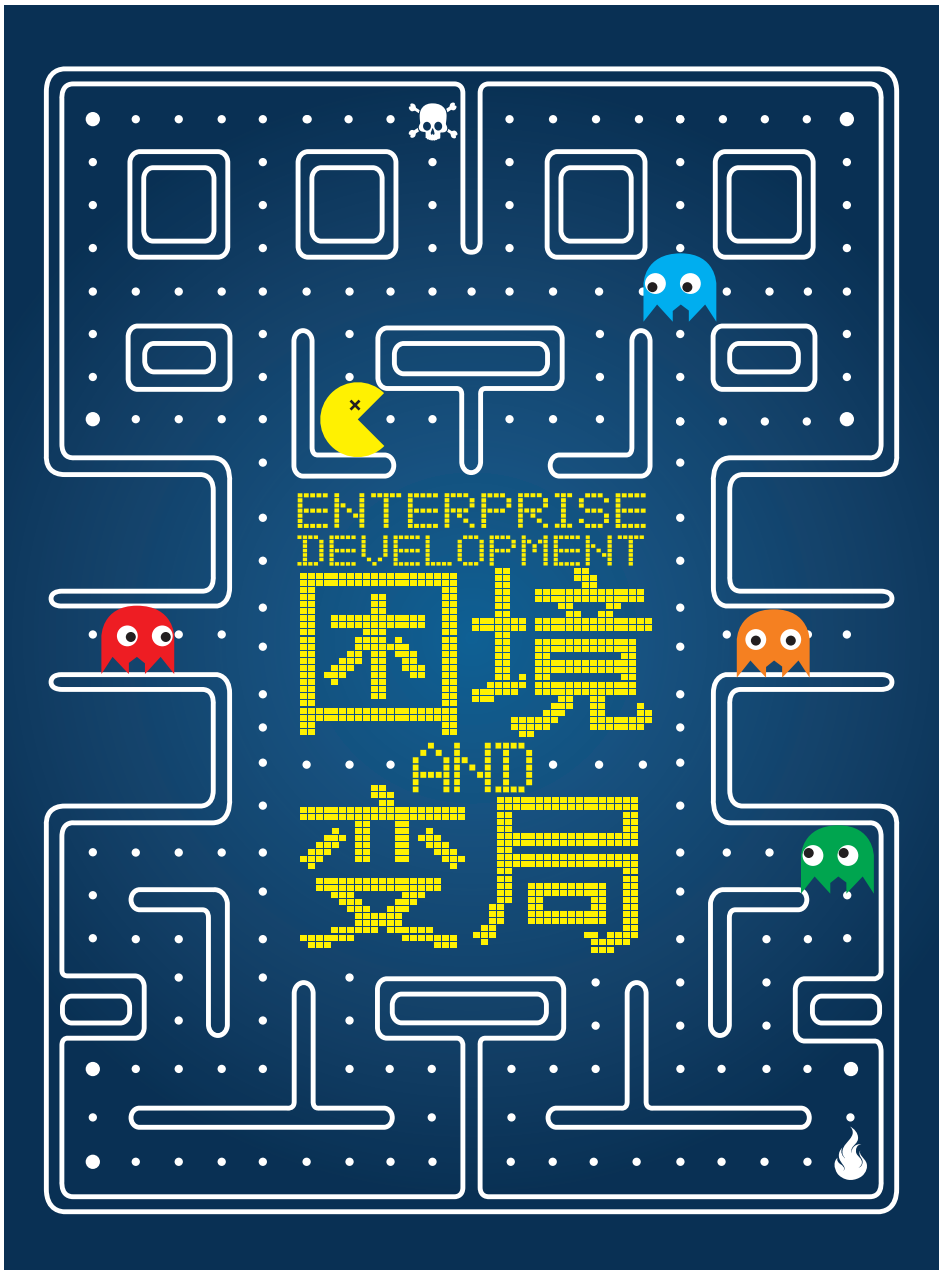
2011.12

PROGRAMMER 程序员



18 Linux Kernel Summit 2011
一位内核开发者的见闻

28 MDCC
2011 中国•移动开发者大会
本次以“移动大变局下的抉择”为主题的开发者盛会，印证了移动黄金时代的来临

34 App Store 式失败八大经验

44 用户体验的跨职能协作

80 硅谷产品管理大师 Marty Cagan
产品探索的意义和流程

90 时间管理的诀窍

100 Twitter 首席工程师 Nathan Marz
如何“打败”CAP定理

114 创建可维护的自动化验收测试

132 Jean-Marie Hullot
Interface Builder神话

135 Marc Benioff
云时代的弄潮儿

ISSN 1672-3252
9 771672 325111 12>
邮发代号：2-665　定价：15元
www.programmer.com.cn　www.csdn.net

Contents

在本刊文章标题处印有此图标的页面，读者可以使用“一拍”（下载地址为 http://itunes.apple.com/us/app/id463494232?mt=8）拍下整幅文章页面，以阅读该篇文章的更多相关内容。
目前仅支持iPhone/iPad，欢迎使用。



P18


P28



资讯

6 资讯
8 新产品新工具
10 外刊速递
13 程序天下事

热点报道

18 一位内核开发者的见闻
——记Linux Kernel Summit 2011大会
20 “用户体验”以人为本
22 2011年Ruby技术大会：松本行弘谈如何选择开发语言
24 乐业背后的玄机
——走进CA Technologies中国技术中心

移动

28 你准备好了吗？
——2011中国·移动开发者大会侧记
34 App Store式失败之八大经验
在本次移动开发者大会上，施炜亮谈的不是成功，而是聚焦于他们研发的失败产品，并一一分享从中获得的经验。这些经验包含更为宝贵的真知灼见。
36 移动设备上的Web App历史以及发展
刘铁峰在本次移动开发者大会上，深入分析了Web App的发展前景。
39 终于轮到开源硬件了
Seeed Studio是目前国内发展的最好的开源硬件厂商之一，廖巍作为嘉宾出席了“2011中国·移动开发者大会”，并撰文全面阐述了开源硬件的发展。
42 圆桌论坛：移动互联网的创业与投资
面对移动投资这片创业红海，投资者如何选择投资方向和创业团队？如何评估在移动领域的投资回报？对移动互联网未来发展方向如何预测？这些都是需要深思的问题。
44 用户体验的跨职能协作
47 泛社交化游戏中的平衡问题
游戏的平衡性往往左右着玩家的心理，决定着用户黏性。通过怎样的玩家策略能够保证玩家在游戏中获得良好的体验?本文将对其进行详细的阐述。



封面报道

由于研发围绕采购者而非最终用户进行，企业软件往往用户体验较差。伴随着互联网的兴起，企业软件的技术渐渐落后。然而，近年来众多新兴企业软件的出现，整个行业正在经历一次大变局。如何摆脱当前的困境？如何面对大变局？精彩内容，请关注本期封面报道。

54 企业开发的困境与变局

59 浅析Java企业开发

64 .NET企业级开发

67 SOA在当今中国企业的发展现状

71 浅析工作流平台

75 基于SOA、ESB、BPM实施敏捷的企业IT系统

管理

80 产品探索的意义和流程

Marty Cagan是享有世界声誉的产品管理专家，曾担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作和经验的总结，分享了探索产品的流程和方法。

85 产品经理要学会说“NO”

88 放任型的教练

——北森CTO张庆化专访

在担任北森CTO之前，张庆化曾任职于Yahoo!和MySpace，拥有深厚的技术积累，而在管理方面，他把自己定义为一名教练。

90 时间管理的诀窍

同样的工作，有些人可以有条不紊、井井有条，高效地完成；而另一些人则整天看起来很忙的样子，工作却迟迟不能按时收工。从一定程度上说，这是一个“时间管理”的问题。本话题邀请了在时间管理方面有话要讲的嘉宾，分享他们时间管理的诀窍。

93 软件企业量化管理的思想方法





云计算

98 云时代的职业挑战

100 如何“打败”CAP定理

CAP定理是数据系统设计的基本理论，目前几乎所有的数据系统的设计都遵循了这个定理。但CAP定理给目前的数据系统带来了许多复杂的、不可控的问题，使得数据系统的设计越来越复杂。Twitter首席工程师、Storm的作者Nathan Marz在本文中通过避开CAP定理带来的诸多复杂问题，展示了一个不同于以往的数据系统设计方案，给我们的数据系统设计带来了全新的思路。

2011年12月刊　总第230期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行主编：孟迎霞　Executive Editor-in-chief：Meng Yingxia
编辑部主任：董世晓　Director：Dong Shixiao
责任编辑：高松　陈博　陈秋歌
Editors：Gao Song　Chen Bo　Chen Qiuge
特邀编辑：方梁　高昂　赵健平　吕娜　卢鹄翔
Contributing Editors：Fang Liang　Gao Aang　Zhao Jianping
Lv Na　Lu Dongxiang
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：朱凯　Art Editor：Zhu Kai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：sales@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Hengsheng Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

本刊文章版权所有 未经许可不得转载
发现装订错误或缺页，请将杂志寄回本刊读者服务部调换

106 软硬兼施，直指海量数据
——解析内存计算技术HANA

108 漫谈基于HTTP协议的SOA架构
基于HTTP协议的SOA架构近来被广泛使用，无论是新浪微博、腾讯微博的开放平台，还是淘宝开放平台，无一不是基于HTTP协议架构进行设计的，本文将详细论述基于HTTP的SOA架构以及亟待解决的一些问题。

技术

114 创建可维护的自动化验收测试
本文原稿首次发表在2009年的敏捷开发实践大会上，彼时是作为敏捷测试研讨分会系列的一部分，此番再刊作者做了一些更新。

119 SDK安装程序卡壳之谜
——兼谈函数的异常出口（上）

122 Y X-RIME: 基于Hadoop的开源大规模社交网络分析工具

程序人生

130 行走在投资界的程序员
——专访千淘资本合伙人李华兵
汽车、极客、足球、开放平台、风险投资、创业，这些是李华兵给自己的标签；汉能投资副总裁、千淘资本的合伙人、车品汇首席执行官是李华兵职业生涯的有力诠释。在本篇中，我们将探寻从技术研发到创业的故事，讲述他跨越两个行业的程序人生。

程序春秋

132 Jean-Marie Hullot的Interface Builder神话
Interface Builder，是用于苹果公司Mac OS X操作系统的软件开发程序，Xcode套件的一部分，于1988年创立。它的创造者Jean—Marie Hullot自称是“一个热爱旅行、充满激情的摄影师”，本篇分享Hullot热爱技术的那一面——创造Interface Builder的过程。

名人堂

135 云时代的弄潮儿Marc Benioff

126 新书上架

128 Geek

136 程序幽默

企业专栏

23 智能电视引发技术新浪潮

26 架构师挑战与武器系列：如何画架构蓝图

50 也说腾讯应用中心

52 工具类应用存在巨大市场

团购

11月初，团购领域标杆Groupon登陆纳斯达克，成为2004年以来最大的科技股IPO。这一举动打开了众多网络公司的“IPO窗口”，包括Yelp、Angie's List和Zynga等在内的公司，也纷纷准备提交IPO申请。

这是Groupon年内第三次启动IPO，因为市场态势不容乐观以及自身运营模式遭遇瓶颈等原因，之前两次被推迟。不过，第三次IPO中Groupon反而受到投资者热捧，得到了10倍以上的超额认购。这意味着虽然外界争议颇多，但在投资者眼中团购模式还是得到认可的。只是原有模式似乎已经走到死胡同，如何在之前基础上，有效演化、转型，才是Groupon大考的开始。或许结合LBS、社交、实时性等时髦功能元素，可以帮助Groupon展开新的作为。受此利好消息鼓舞，反观国内团购网站，是否也能如Groupon一般风光上市呢？业内专家的看法似乎一致地悲观。原本定于11月14日上市的拉手网取消了IPO，无限期推延，原因众说纷纭。

自2010年初，第一家团购网站上线至今，国内团购网数量已达千家以上，互联网巨头或者普通创业者都急冲冲杀入战场，“千团大战”至今硝烟未尽。如此浮躁、跟风，问题自然层出不穷，极低的毛利润、过度透支商家资源、一味山寨不知创新，团购模式给了人们疯狂烧钱、恶性竞争的不良印象。

如今，Groupon跌跌撞撞地IPO之后，被资金链断裂、欠债、裁员等问题困扰的国内团购网站是否能学得有模有样，真的很难说。

SAP中国商业同略会暨全球技术研发者大会



11月15-17日，SAP中国商业同略会暨全球技术研发者大会在北京国家会议中心举行。近万人的参会规模，只能用“这是个大会”来形容比较贴切。SAP中国商业同略会源于SAPPHIRE，也就是常说的“蓝宝石大会”，偏重市场化，通常在国外举办，今年第一次来到中国。全球技术研发者大会（TechEd）关注的则是知识的传递。两个大会在中国首度联手，彰显了SAP对于中国市场的重视程度。大会围绕着云计算、移动、内存计算展开。

这次大会有几个亮点还是值得称道的。首先是海尔集团首席执行官张瑞敏结合自身实践，谈了海尔在IT信息管理方面的探索，可以说是从油水分离的阶段，到了现在水溶交融的阶段。接着是篮球巨星姚明现身说法，分享了“青年•教育•未来”的话题。而农夫山泉CIO胡健则在演讲中，列举了内存计算技术HANA与Oracle数据集市的比较，将Oracle批判得体无完肤。另外，SAP公司联合首席执行官孟鼎铭在大会上表示，2015年之前SAP计划在中国投入超过20亿美元，用于解决方案创新、解决方案交付、支持解决方案和推动中国IT生态系统发展。如此大规模的投入，无疑为波澜不惊的企业软件市场注入了一针强心剂。但最终结果如何，还有待时间来考验。

苹果应用商店

11月18日，中国的苹果用户发现，App Store首页底部，已经添加了充值功能，支持国内各大银行。这表明在iPhone正式进入中国两年后，苹果全面铺开本地化运营的进程。对于饱受黑卡、破解越狱之苦的应用运营商来说，这无疑是个大惊喜，上述问题也将因此得到极大缓解。在本次币种障碍消除之前，中国用户购买苹果应用软件必须使用美元支付。这一定程度上使得用户购买意愿下降，破解、越狱行为泛滥。

今后，在苹果中国区的收入将以人民币形式分成，简化了结汇流程。消息当天，很多应用下载量都有了成倍的增长。

不过，外界对苹果此举也是疑虑重重。苹果联手的首信易支付（PayEase）尚未获得央行第三方支付牌照，同时购买苹果应用软件属于电信增值服务，按中国现有法规，外资不能独资经营此项业务，只能以合资企业的方式经营且最高控股比例不得超过50%。这个风险可能会由苹果在中国注册公司规避，但如果App Store在中国运营，其中涉及图书、音乐、游戏等内容都需要相关部门的许可，对开发者来说若不能取得许可证，应用有被下架的可能。

虽然目前此事还有诸多争论，消费者是否会因此改善消费习惯，开发者是否会因此获得更优质的收益，究竟会带来怎样的影响，短期内还很难显现出结果。但有一点很容易确认，苹果这样的超级公司对于中国市场的潜力，乃至中国版权保护的未来，还是有相当的信心的。

“史蒂夫和我共同努力，创造麦金塔计算机。（微软）投入较多人力在上面，研发麦金塔的主要软件。”

—— 56岁的比尔·盖茨接受美国广播公司Christiane Amanpour访问，一改他先前在乔布斯逝世后，对这位苹果共同创办人坚定不移的赞美。英国《每日邮报》报道，盖茨如今采取含蓄反击。

“这部分损失，苹果公司是不会承担的。”

——面对黑卡泛滥，坏账率激增，国内iOS平台生态环境恶化，热门游戏《捕鱼达人》开发者、触控科技CEO陈昊芝如是说。

“在苹果手机上发现的中毒案例，不会像在Android手机上这么多。”

——数据表明，7月以来Google Android手机操作系统遭到病毒、间谍程序等安全威胁的次数增加近6倍，Juniper网络公司主导此项研究的霍夫曼对此评论。

“还在想招儿，这个确实有难度。我们确实把这个责任交给了腾讯和金山来干。”

——被问到如何在浏览器领域战胜360，王小川如此表示。从渗透率的角度，360浏览器市场份额是搜狗浏览器的两倍多一些。

“一般来说，用户对隐私保护都有自己的期望，我们要尽可能地满足他们的这种期望。”

——W3C跟踪保护联合主席Matthias Schunter如是说。国际标准组织W3C提出了一些新的隐私标准，这些标准允许用户对于他们的隐私拥有更多的控制权，而且得到了Facebook和Google等主要网站的支持。

“我认为分拆公司无法创造出根本性的价值。”

——微软首席执行官Steve Ballmer在公司年度股东大会中表示，微软的优势在于能够驾驭整个公司的协同作用。

“只要我们保持增长超越对手，登顶是迟早的事情。”

——超越惠普，成为全球PC老大，联想董事会主席杨元庆对此也毫不避讳。

“以前全球每家大企业都必定有两套数据库方案，用于OLTP和数据仓库的关系型数据库以及用于商业报告和商业智能的数据库。现在企业们正在增添第三套方案——NoSQL。”

——10gen CEO及联合创始人Dwight Merriman接受采访时如是说。

CSDN十大资讯

2011年10月26日~11月26日

01 Siri严重威胁搜索领域业务

作者 / longago

Eric Schmidt向美国反垄断委员会致信阐述Apple的Siri将影响Google的搜索业务，称Siri不仅能执行常规任务，其最大特点是语音搜索。Schmidt甚至认为Siri是“Google杀手”。此外，Schmidt还表示Google在垂直搜索、社交网络、商业软件、移动导航等领域都有很多竞争对手。

02 Windows 8十大安全漏洞缠身

作者 / 范平

Windows 8是否值得信赖？黑客曝露了Windows 8有欠考虑的安全细节。本文介绍了应用程序安全级别；漏洞百出的Flash平台；电子邮件、共享页面和网页链接；用户习惯；平板电脑的安全盲区；IE浏览器；Windows Defender以及Hyper等十个安全隐患，提醒大家保持警惕。

03 阿里10亿投一淘，两年内不考虑盈利

作者 / 徐洁云

阿里巴巴集团宣布投资一淘网10亿元，帮助网站获取外部流量，此外，将淘宝直通车、淘宝联盟等业务划入一淘。一淘网宣布向所有电子商务网站开放淘宝、支付宝用户资源。公司总裁吴咏铭表示，目前正在努力做大，还不是考虑盈利的时候。

04 美国企业家发布《中国创业报告》

作者 / 书聿

美国企业家Bowei Gai发布了《中国创业报告》，介绍中国科技创业环境，包括竞争毫无节制；用户体验与美国截然不同；创办公司外部条件复杂、技术环境落后等等。Gai认为创业者有必要亲自到中国了解情况，重点是北京，并且必须懂中文。

05 JavaFX开源计划细节泄露

作者 / Paul Krill

Oracle Client Java首席架构师Richard Bair透露了JavaFX的一些开源计划，暗示用开源代码取代所有封闭代码。Oracle希望JavaFX能够为下一代Java客户端工具包做出贡献。JavaFX目前包括6000多个公共API和组件。在适用范围上，JavaFX目前还落后于其他一些富媒体技术。

06 优秀程序员的首要特性：判断力

作者 / Tammer Saleh

作者说很多公司在招聘Ruby on Rails程序员时，往往忽略了一名出色开发者所应具有的首要特征：判断力。没有判断力的开发者会陷入精心设计最简洁解决方案的陷阱中，并不知道什么时候才超出他们的能力，所以总会在整个项目中留下代码炸弹。



07 Adobe停止开发移动版Flash

作者 / 唐风

Adobe宣布停止为移动浏览器开发Flash Player，这表明该公司基本上承认在与竞争对手Apple长期以来的网络标准“战争”中落败，HTML5已成为移动开发者偏爱的技术。停止开发移动版Flash Player后，Adobe业务重心将转向为使用HTML5工具开发网站的人员出售开发工具。

08 Google正在建立自己的Facebook

作者 /卢鑫

Mark Zuckerberg采访中对近年来科技巨头的大战不屑一顾。Zuckerberg说Amazon和Apple是盟军、是战友。然而，从某些角度来说，Google与Facebook存在竞争关系。Google目前正在建立自己的Facebook。Zuckerberg还坚决表示，Facebook永远不会推出自有品牌的游戏。

09 传马化腾曾想收购UC但遭拒绝

作者 / 谢晓萍

10月24日，UC指责腾讯发布的手机浏览器市场份额报告与事实不符；11月3日晚，马化腾给俞永福发了一条短信，希望UC可以取消第二天“声讨”腾讯的发布会，但是遭到拒绝；11月4日，UC宣布，将以不正当竞争为由起诉腾讯。

10 小米是最后一件事，投资只看熟人

作者 / 郑峻

雷军表示，小米是自己创业做的最后一件事，只想做一款出色的手机；只投给相信的熟人，更看重创业团队而非创意，也不在意投资的公司是否上市。小米、UC、多玩与腾讯的竞争以及金山和腾讯的合作是两码事，自己的角色并没有什么尴尬。

2012年10大技术发展趋势

假如要选出明年将会影响到企业基础架构的10大技术发展趋势，Gartner认为还是必须首先考虑虚拟化，然后再考虑其他趋势，如社交媒体的影响、节能降耗问题以及扁平网络等。

在Gartner举办的年度IT研讨会上，Gartner执行副总裁兼基础架构团队首席研究员David Cappuccio指出，Gartner所选择的这10大趋势表明，IT正在改变过去很多的传统看法，这些变化将会影响到IT未来的运营。

01 虚拟化

虚拟化将会最终促使更多的企业把IT视为一项业务。朝着构建一个可以不断更新的战略规划方向努力。

02 大数据

非结构化数据将会在未来五年内出现80%左右的增长，分析以及其他监控重复数据的系统也可能带来收益。

03 能效

能耗问题已经上升到了吞噬企业产业链的地步，IT部门需要搞清楚哪些计算资源正在做什么，哪些工作会浪费能源。

04 环境感知应用

环境感知计算将会超越商务智能应用，真正产生一种统一通信环境，把来自社交网络和移动设备的各种数据组合在一起。

05 员工再培训

制订让员工对工作感到振奋的计划。如何留住员工，并帮助其再学习、再培训，对于企业的稳定发展来说至关重要。

06 社交网络

人与人方便地聚合在一起，以集体的声音表达对企业产品/服务的意见。忽略社交网络对企业发展来说是致命的。

07 消费化

开发出移动新应用解决用户问题，但它们无法替代桌面应用。需要制订安全、定义完善的战略发挥移动的优势。

08 多核

16核系统也将在未来两年内流行开来，但这会增大整个数据中心的能耗。IT部门还必须解决性能/许可证之类的问题。

09 云计算

云计算的好处是与生俱来的弹性和扩展性。IT应用越多工业化，越少定制化，从云计算获得益处也就越多。

10 Fabrics

服务器、存储和网络系统的垂直整合，这种整合为数据中心资源有效和动态优化共享奠定了基础。

Flurry为应用开发者推新工具，提高用户参与度

iOS应用商店有50万个应用，Android应用市场里也有30多万个，要想让用户注意到你的应用，是一件十分困难的事情。进入应用排行榜中是吸引新用户的方式之一，但要停留在那里并不容易。移动应用分析公司Flurry新推出的这款工具，是一款让iOS开发者尝试激励那些只下载应用但可能不使用的用户的工具。Flurry已经明确地找出了在一段时间内应用使用率下降的原因，并且在这个时候进入这个拥挤且不断成长的应用工具市场，其目的就是要帮助开发者增加其应用的使用率。

用户可以免费获得Flurry的分析数据，并决定面向哪些用户打广告。当用户使用其他应用时，Flurry会向他们显示广告，提醒他们你的应用有更新，并提供一个链接，用户可以点击它直接打开应用。



开源商业智能解决方案Pentaho 4.1发布



该版本在数据分析性能、可扩展及可靠性方面做出了重大改进，并且支持分布式In-memory缓存系统。Pentaho是世界上最流行的BI解决方案。它是一个基于Java平台的BI套件，包括一个Web Server平台和几个工具软件：报表、分析、图表、数据集成、数据挖掘等工具，可以说包括了商业智能的方方面面。Pentaho BI同时提供了社区套件和企业套件两个版本，社区版完全免费。

Pentaho 4.1包含了对Infinispan/JBoss企业数据网格及Memcahed缓存系统的数据分析支持，同时也可以扩展到其他系统。另外的改进则包括全新的性能辅助调整、支持对Apache Hive及EMC的Greenplum数据库的本地SQL代码自动生成等。

Joshfire开发框架

随着智能手机和平板电脑的不断普及，开发者往往需要针对不同的设备开发产品。Joshfire是一个支持多种设备的开发框架，仅采用HTML5和JavaScript等标准，可以帮助开发者迅速开发本地专用的网络应用，用于浏览器、Node.js、桌面电脑、智能手机、智能电视以及联网设备。



SoybeanMilk Java MVC开发框架发布

SoybeanMilk是一个简易、友好且零侵入的Java MVC实现框架：它几乎没有学习成本，只需要熟悉JSP和Servlet技术；也不需要遵从任何代码编写模式；代码中找不到任何这个框架的踪迹。使用它，仅需要编写少量的XML代码，就可以轻松地将任何Java方法发布为Web应用。 另外，它还支持RESTful风格；内置强大且易扩展的对象转换器，可以自动将请求参数转换为复杂类型的对象；可以很容易与Spring、Velocity等当前流行的框架整合。

SoybeanMilk-1.0-b4做了较大的改进，包括：增强泛型类型支持，框架现在支持自动将请求参数转换为调用方法的泛型类型参数等。

Mozilla发布HTML5工具包Popcorn.js

Mozilla发布了一个HTML5媒体工具包——Popcorn（爆米花），通过Popcorn，视频制作人员可以利用JavaScript使用各种社交媒体的实时信息，实现可视化信息，并直接置入在线视频，这应该是一种新的交互电影雏形了。广告业的开发者可以好好研究一下Popcorn.js和PopcornMaker。

HTML编辑器xhEditor v1.1.11发布

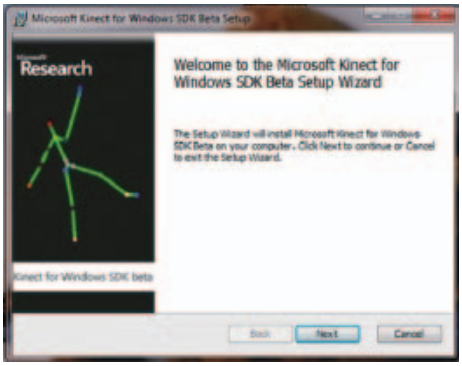
xhEditor是一个基于jQuery的简单迷你并且高效的在线可视化HTML编辑器，基于网络访问，并且兼容IE 6.0+、Firefox 3.0+、Opera 9.6+、Chrome 1.0+和 Safari 3.22+。 其新版本v1.1.11，更新内容如下：添加Adobe Air平台的兼容支持；修正了某些国内Webkit内核的马甲浏览器无法打开iframe模式窗口的修正等。

GitHub推出Enterprise服务

GitHub宣布推出了GitHub Enterprise服务，为公司提供自托管的代码存取及版本管理服务。根据Github官方博客，GitHub Enterprise包括了Github之前的所有主要功能、提交历史、代码浏览、比较视图、推送请求、问题追踪、Wiki、Gist共享代码段和团队管理等。此外，还提供了更强大的API以及一个更漂亮的Web界面。在GitHub Enterprise上，甚至可以在自己的服务器本地运行这些功能。

不仅如此，GitHub Enterprise还支持LDAP和CAS，可使用企业已有的认证服务。提供所有数据的完整系统备份机制，自定义的服务器端Git Hooks，提供HTTPS访问支持等安全措施。

微软升级Kinect开发工具包



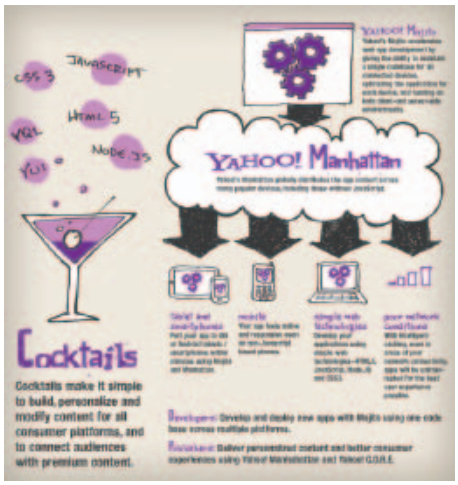
微软推出了其体感控制设备Kinect的Windows开发工具包的Beta 2版本，该版本中添加了对于Windows 8的支持。微软在Beta 2版本的自述文档中称：“开发Kinect应用本质上和开发其他Windows应用一样，不同的是该SDK支持Kinect感应器的相关功能，比如彩色图像、深度图像、音频、骨骼动画数据等。” 除了添加对Windows 8的支持外，SDK Beta 2还包括如下改进： 显著改善骨骼跟踪，包括更快的骨骼框架提交和多线程/多核支持；整体提升了精确度，可产生更精确的跟踪；当使用2个Kinect时，开发人员现在可以指定其中一个用于骨骼跟踪；添加对检测和管理设备状态更改的API支持，如设备断开、设备接入、电源断开等；支持构建64位应用程序。此前，只能构建32位应用程序；引入一个新的C#示例：KinectAudioDemo等。

另外，微软此前宣布，将向企业用户出售Kinect for Windows SDK的商用版本，预计于2012年初发布。

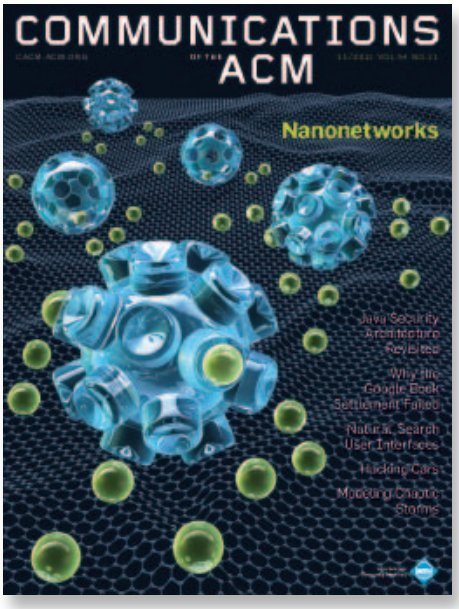
Web开发框架Mojito及其托管平台Manhattan

Yahoo!在其开发者博客发布了一项新的Web方面的技术——“Cocktails”。Yahoo!认为该技术将“深刻影响开发者社区”，并且正式介绍了两个Cocktails技术：Mojito（一个JavaScript的Web应用程序框架）和Manhattan（针对Mojito程序的一个托管平台）。

Mojito是一个JavaScript Web应用框架，使用其开发的程序可同时运行在浏览器和服务器端。即使浏览器中没有启用JavaScript支持，Mojito程序仍然可以运行在服务器端，全部使用一个单独的代码库。



Manhattan则是一个针对Mojito程序的服务器端的JavaScript托管平台。它对Node.js进行了扩展，以便提供必要的故障隔离和容错功能，增加程序可扩展性、可用性、安全性及性能方面的保障。Yahoo! Manhattan为开发人员提供了一个简单的界面，可用于部署、卸载及管理他们的Mojito程序，通过借助Yahoo!遍及全球的技术基础设施，例如Yahoo!的内容分发网络（CDN），可以有效减少用户的网络延迟。



Communications of ACM

2011.11

搜索的自然用户界面

最新的趋势表明，用户界面朝着更加“自然”的方向发展，包括人机自然交互、手势控制、取代键盘的语音输入、取代文本的视频阅读方式，以及IT的社会化趋势。这些趋势同样影响了搜索用户界面。通过使用先进处理技术对大量行为数据进行分析，未来的搜索可以帮助用户定位其他人来提供最佳答案或建议，交互方式也更加接近于自然对话，信息的输入输出方式都不再仅仅是静态文本。实现如上所述的自然用户界面是一个长远的目标，但最近的技术发展让它越来越接近现实。

语音输入

得益于移动电话的普及、语音识别技术的发展和触摸界面的推广，语音搜索将在今后几年内得到更好的应用。语音搜索的下一个趋势，将是自然对话式的输入输出，就像最新苹果手机上应用的Siri系统，它是一个通过语音交互的智能个人助理。

社交搜索

社交搜索的三个主要趋势是协作、众包和社交问答。

经验观察发现，人们经常会进行协作搜索，但支持多人协作的新搜索工具最近才被开发出来，这主要受益于一系列实时分享工具研究的进展。其中一项进展是，通过在协作搜索中引入评级算法，从而允许用户进行相对独立的搜索，但同时其他团队成员的搜索行为会对其产生实时影响。除了同时同地进行的“协作”搜索外，“众包”搜索通常是通过对信息进行评价和分享来实现的。

在社交网络中，人们的提问通常带有主观色彩（社交问答网站Aardvark上随机取样的1000条查询中，65%都带有主观色彩），这些问题很难在互联网页上找到现成的答案。哪种用户界面更适合这种更具有社交性质的搜索，还没有定论。Yahoo!和Google先后推出了MyWeb和“+1”，在搜索结果中显示用户社交网络对各条目的评价情况。也有研究者开发问答系统，为提问者寻找潜在的专家来解答问题。

自然语言查询

人们进行搜索时使用的查询字段长度逐渐增加，这表明了用户希望更完整、明确地表达自己想要获得的信息，最终搜索字段可能会发展成一个完整的句子。随着语音识别更加快速和更加准确，语音搜索也可能会使用完整的句子。语言分析、使用结构知识库的命令语言，以及交互的改进，将会促进对话交互方式的发展，就像前面提到的Siri一样。

视频内容的进展

越来越多的证据表明，普通信息消费者中，视频和音频内容超过了文本内容。调查显示，52%的美国成年人观看在线视频。2010年，视频分享网站Youtube成为全美访问量第5的网站，其搜索量超过了Yahoo!和Bing搜索。视频迅速成为一种方便的沟通方式，而移动设备让视频的获取变得更容易。相对于文本来说，视频的相关处理技术还远远落后，但视频快速编辑的更佳工具将会出现。如何将视频以有意义的方式进行总结，并对其进行搜索，也在研究当中。

再探Java安全架构

1995年，Java平台JDK 1.0中只包含了非常简单的“沙盒”安全模型。1996年，Li Gong加入Sun微系统公司JavaSoft部门，对Java的安全架构进行重新设计。该安全架构包含在1998年发布的JDK 1.2中，如今被应用到了无数系统和设备中。他撰写的这篇文章，从设计和工程的角度，重新回顾了Java安全架构中一些棘手的技术难题。

他总结了Java安全架构的4个特点：可用性、简单性、充分性、适应性。在最初的JDK 1.0中，安全机制只针对两种具体的情况——“沙盒”内和“沙盒”外的代码，这种看似简单的架构导致了设计的复杂性、代码的脆弱性以及层出不穷的安全缺陷。在JDK 1.2中，安全机制是通用的、系统的、简单的，这增强了平台的健壮性和可用性。

安全架构的设计目标是保证Java代码能够如预想中运行，而不产生意料之外的副作用。这个目标包含3个部分。

首先，要保证代码的有效性。通常，我们书写程序的Java源代码，然后编译成平台独立的Java字节码，然后在Java虚拟机上运行；也可以直接书写Java字节码。但不管以哪种方式，得到的代码都有可能是不可信的。因此，JVM必须能够判断一条字节码是否有效可行。从根本上来说，这是一个输入有效性检验的问题。其次，要保证代码的行为能够和预想中一致。最后，要预防代码产生意外的错误行为。为了解决这个问题，JDK 1.2中采用了最小特权原则。在这个领域还存在着很多棘手的问题，其中最大的问题就是最小特权计算能否有效进行，特别是在安全参数非常复杂的情况下。另一个长期存在的问题，是安全策略管理和部署的实际应用。

另外，还有一些需要注意的技术问题。比如NULL的使

用、运行中静态代码到动态域对象的转换；Java无法保证连续指令的顺序执行等。

OCaml

函数式编程已经有了数十年的历史。早在计算时代诞生之初，就出现了Lisp这样的函数式语言，静态类型函数式语言OCaml和Haskell是这个家族较新的成员。函数式编程有着巨大的影响力，但从来没有真正地成为主流。很久以前，它有过辉煌；最近几年，函数式编程有复苏的迹象，但也仅限于讨论，而非应用。人们错误地认为：函数式语言只局限于某些特定应用，包含一些可以引入到其他语言的有用观念；但命令式语言和面向对象的语言适用于大部分的软件工程。本文的作者以自己10多年来应用OCaml进行产品开发的经验，说明了函数式语言，特别是静态类型的函数式语言，如OCaml和Haskell，是优秀的通用编程工具。

作为一种函数式语言，OCaml的关键优势在于：

简洁：使用OCaml，可以创建比Java或C#更简洁、更易于理解的系统。对于看重可读性的机构来说，这是一个巨大优势。

缺陷检测：这种书写系统捕捉缺陷的效率令人吃惊。如果你的代码能够通过书写检查，它基本上就不会存在错误。

性能：OCaml的性能与Java相当，甚至更优，接近C或C++。除了优秀的代码生成器外，它还拥有一个增量型的废料收集器。

不“纯”：OCaml允许可变状态。“纯粹”的函数式语言不允许任何可变状态。但可变状态是编程的一个基础，必不可缺。

OCaml并不是完美无瑕的。虽然OCaml拥有一个大的社区，为它提供了丰富的库，但与Python、C和Java的库相比，还是相形见绌。同样地，IDE、性能分析、调试器等相关的开发工具也不够成熟。OCaml的另一个局限性在于对并行的处理，支持多进程Ocaml编程的工具也还在完善中。虽然在实现的细节和该语言的推广上还需要加以努力，但从根本上说，Ocaml本身并不存在局限性。



Wired
2011.11

Facebook、Spotify和音乐的未来

即使是Facebook CEO Mark Zuckerberg两年前也没有注意到Spotify，然而当两个公司被联系到一起将是不可避免的。欧洲的音乐服务已经吸引了数百万的用户，这得益于一种商业模式，允许音乐爱好者即时、免费地听音乐，更重要的是，使得人们更容易彼此分享音乐。从这个角度来说，非常符合Facebook的世界观——共享价值胜过其他一切。这也正是为什么Fackbook前总裁和Napster联合创始人Sean Paker在2009年发现Spotify，就立刻告知了Zuckerberg。“我从没听说过Spotify，但Sean有一天跟我提起它”，Zuckerberg说，“我想这家伙真的做了一个很酷的音乐产品并且知道如何把社交元素整合进去。”当天，Zuckerberg就更新了自己的Facebook状态：Spotify如此棒。

这个简短的祝福，预示着整个音乐产业的转折点。最初的Napster（允许用户免费下载任何歌曲）已经死掉将近十年了，但它仍然困扰着各大唱片公司。虽然2001年它在法庭上被惩罚，但他教会了一代人，音乐应该使用鼠标点击，而不是用钱。唱片公司对此态度截然想法：人们习惯了在网上分享音乐，将永远不会再为其付费了。他们对数字音乐一直充满恐惧。生怕Napster再生毁掉他们剩下的买卖。

今年9月，在Zuckerberg说“Spotify如此棒”两年之后，Facebook宣布了一个野心勃勃的计划，让用户更快更容易分享音乐（大部分是免费的）。Facebook正在Spotify紧密合作（还有其他十几项服务和潜在合作方），现在Facebook的用户将看到，他们朋友听的歌，编辑的播放列表和新听的乐队，而且只需要点一下鼠标，就可以听到所有的音乐。

Facebook的音乐服务只是新Napster的一个例子——分享音乐到本地存储到在线音乐流分享。事实上，在过去一年中，每一个主导的互联网公司，包括Apple、Amazon和Google，都已经憋足了劲儿开发自己的流媒体音乐服务，试图改变大家购买和听音乐的方式。那些小公司（早期的Rhapsody和Pandora）已经在此领域奋斗多年。现在又有新的成员加入，比如Rdio、MOG和Turntable。

这些公司一起正在撼动顽固抗拒变化的行业。自从音乐成为文件，音乐世界就从未被有效地管理过。流媒体订阅服务催生了一个更大的动荡：歌曲成为连接。随着新闻推、电子邮件或是Facebook的个人资料一起被播放。这也正是有趣的开始。

如今EK把无限制地听音乐当作理所当然的事儿。但他吸

取了Napster的教训，采用流媒体的方式，并且在此过程投放广告，从而与唱片公司分成。Spotify不是第一个允许用户听流媒体的音乐服务，但过去十年里，向用户收费的服务总是没有太大起色直到EK推出了自己的服务。在此期间，EK指出Apple的iPhone和Facebook成为互联网巨人，一起定义了两个新领域：移动互联网和社交网络，并且EK认为这将帮助他建立梦寐以求的服务。智能手机意味着人们可以在任何场合听歌，而不必被拴在电脑旁边；社交网络意味着可以从朋友那里获得音乐，而不是数字商店或者盗版者。

EK正在创造一种流媒体服务，能够超越iTunes。Spotify也如同Apple的线上商店一样简洁性与功能性并存。它就和iTunes一样快，要知道RK的用户并不是从本地寻找歌曲，而是从网上的超过15万首歌，这绝非易事。他们需要确保95%的音乐在200毫秒内被发送到用户。这感受就和在自己电脑里听歌一样。



Technology Review
2011.11

云，势在必行

在Facebook、Google，乃至有互联网之前，科学家们就有把计算当作资源的观点。在他们看来每一位用户只需付费给他使用的计算量，用户就可以拥有巨大系统的全面支持。

与其自己买计算机系统，公司、个人甚至政府能够公共的计算设施，不间断地提供可替换的部件、计算能力、数据存储以及通讯能力。如果部分出现故障或者需要更新，程序和数据可以自动转移。多级安全机制可以防止用户之间的相互干扰。比起分布在不同企业机构的单独的计算机，这个庞大的系统能够使硬件利用率更高和管理人员效率更高，花费更廉价。

“虚拟化”意味着可以随时产生需要的资源。像Amazon的服务提供商，可以大规模地提供付费实用的计费模式。这一转变的后果是深远的。最明显的是，今天很少需要为企业购买除员工的PC和笔记本电脑以外的电脑系统。如果需要邮件服务器或者高性能计算机群计算机，他们可以节省下钱购买云服务，而不是自己购买系统。

让我们考虑一下在公司里处理电子邮件的经济学。今天一部可以接受存储路由入门级戴尔服务器成本低于300美元。但你还要这台服务器增加Windows Server软件、第二块用于冗余的硬盘、Microsoft Exchange Server 2010，并雇用一位系统管理员去管理这些邮件。此外每一位雇员使用软件的License需要35美元。为一个50人的部门将支付高达3250美元的牌照费用

相比之下，如果选择微软基于云的服务，每人每月10美元，提供无限的存储空间。表面上看一年6000美元的支出不是一个更好的选择，但如果自己雇人来管理邮件服务器，保持安全升级，还需要支付电费等隐性成本。云服务则可以备份数据在多个地点，并连接到手机和群组日历。最重要的是，你只需购买你所需要的。

尽管云计算有这些优势，很多企业仍会表示不会采用用，因为对它的安全可靠没有充分的信心。的确如此，Google Docs停运和Amazon今年4月面临的尴尬处境，使得一些用户丢失了服务和文件。但我们要知道，公司自己管理的服务器也会宕机，而且不会少于几小时。更重要的是，Google和Amazon会为事故负责，它们都是上市公司，会发出详细报告告知发生了什么事情、有多大问题以及它们为防止这类事情再次发生正在做什么，而公司自己的IT团队可不会做这些事情。

然而，很多公司都对放权给云计算感到不安，今年早些的一项研究报告显示，大多数受访者首选使用云计算的方式是私有云。对于终端用户来说，由公司自己而不是第三方建立的私有云在使用上并无不同。但建立私有云并非一个小工程，私有云需要有云计算所有的能力——虚拟化基础设施、数据中心、冗余备份、异地备份、冷却设备和电源等。对于单一组织来说，成本是巨大的，根本无法获得云计算的最佳效益。运行一个私有云基本上等于成为云托管服务提供商，除非从不在乎销售产品的成本，私有云可能只对信息敏感的组织有意义——军队或者银行——这些数据不应该接进公共互联网。

还有一些领域，以云计算为基础的服务似乎没有那么大的优势，那就是桌面的应用程序，比如文字处理、表格、演示软件和日历。Google和微软的确也提供了基于云的办公应用软件，但桌面版本更快、更灵活、更容易使用。更重要的是，在一些没有互联网接入的地方也可以编辑文档。然而当你自己的电脑丢失了，或许还要回到云上备份回来。

总结一下，尽管互联网上的每一个组织都或多或少用一些云服务，但他们应该用的更多。如今经济成长的规模让人诧异，还去买一组服务器的家伙是该好好算笔账了。

双通道将遭遇挑战

实施双通道并不能免除对用户网银安全的保障责任，否则会引发新的综合问题。

11月安全会议纷至沓来，移动互联网国际研讨会安全分论坛、云安全联盟高峰论坛、RSA2011信息安全国际论坛、OWASP 2011亚洲峰会等先后在北京举行。信息安全的布道者、从业者和学习者们穿梭其间，成为一道独特的安全交流风景。

这些会议的主题基本集中在移动互联网、云计算和物联网等新兴安全领域上。我相对更加关注移动互联网的安全问题。会议期间，关于短信能否充分保证网银和交易安全问题的讨论让我印象深刻。这场讨论由我介绍的关于PC木马Zeus与手机木马Zitmo合作攻击的问题引发。我认为，由于与场景有关的前提条件存在差异，不能按照传统密码学的双通道安全的观点来认定PC交易、手机确认或告知的模式是安全的。

传统的双通道是在以人为端点的场景下构建的，其核心是通过不同的通道分别发送密文和密钥，以提高攻击者监听的难度。在这个场景下，如果不能同时监听两个通道，那么通信是安全的。

此前我们也是这样做的，如一些基本的保密手册，要求在不能使用PKI加密体系的情况下，采用邮件发送密文、短信发送密码的要求。这种策略下，只有对PC网络和移动网络同时嗅探才能还原通信过程。在低交换时代，两条不同的链路，甚至同一条链路上不同的协议和加密方式（比如加密邮件和安全的IM），也曾被视为双通道。

但双通道的安全并非没有前提，它需要：端是安全的、两个通道彼此是隔离的。

而在实际场景中，节点往往比信道更脆弱。节点如果作为信道的一部分，信道可能就是不安全的；节点如果作为端的一部分，那么这个端本身就是不可信的。

在当今智能手机的时代，PC和手机之间的交换是频繁的，PC端具有通过同步接口远程安装或者写入的权限，也可以通过同步接口进行各种数据的读取。而手机端通过autorun.inf等机制交叉感染PC的病毒亦早已被发现。因此两个通道间并不具备有效的隔离性。

将移动网络作为第二通道的设计，一定程度上是由于GSM在链路层的窃听难度。但由于GSM监听设备的廉价化，完全可能出现在PC通道上触发手机通道信息认证、在第二通道上定向监听获取的攻击方式。

从Zeus和Zitmo合作攻击（以及SpyEye与Spitmo合作攻击）的问题来看，对双通道的另外一种威胁是，即使两条通道之间并不存在交叉渗透的情况，当攻击方在两套通道上都掌握足够多的资源时，也可以寻找一定的对应条件，找到在两个通道的对接点。因此犯罪集团如果掌握了足够多的双通道节点，就会尽力进行双通道之间的信息聚合。

当然，我们对安全要采用聊胜于无的态度，双通道方式毫无疑问提升了攻击者的成本，这就说明它是一个值得继续推广的方法。但我们需要避免误以为双通道是一个百分之百保险的方式。特别是如果认为银行实施了双通道就可以免除其对用户网银财产的安全保障责任，这就会引发新的综合问题。



肖新光
网名江海客，安天实验室首席技术架构师，研究方向为反病毒和计算机犯罪取证等。微博：weibo.com/seak。

技术和数据驱动下的电子商务

电子商务网站的本质就是互联网，这是由技术和数据驱动的。

最近，在中国电子商务大会暨电子商务博览会上，好乐买总裁李树斌在会上发表题为《10亿，突破技术瓶颈》的演讲。在演讲中，李树斌说：“在中国，我实在找不出一个以技术为驱动的案例，但中国都是以亚马逊为榜样的。亚马逊从第一天开始就定义自己是一家科技公司，这是我从来没有想到的事情。”看到这段话，使我想到了国内其他几个

大的门户网站和电子商务网站：京东商城、当当网、新浪、搜狐、网易、淘宝、人人网、开心网等。在《程序员》杂志2011年第一期的专栏里，我也介绍了几个姗姗而来的国产企业开源产品，淘宝、人人网和天涯都在基于自家的应用，做自己的开源产品，并推向了社区。但京东商城，虽然主营业务还是自己在做，但也是以成为平台型网站为自己的核心目标。不过我们却看到京东商城在11月1日的促销活动中暴露出其技术上的短板。即使在次日“增加三倍服务器”后，技术问题仍没有得到解决。刘强东的微博上，更是爆出拿着两把匕首，在会议室，等信息部的同事们喝咖啡的照片。当然这是一个出离愤怒后，无可奈何开玩笑的照片。国内做事就是这样，无论做任何事情，都是商业利益驱动，先解决怎么卖的问题，再解决怎么做的问题。销售的地位总是排在技术之前，也就是用户需求驱动下的技术，而不是技术驱动下的销售与营销。

京东商城和当当网的技术架构，都基于.NET。当然，在这里，我也并不是要讨论.NET和J2EE或者LAMP究竟哪个架构好这等讨论了几百年也不会有准确结论的话题。我想说的是，基于开源技术架构的公司，无论是软件公司，还是电子商务公司，其以后的基因才可能是开源和开放的，才可能会用开源和开放的眼光去看待营销和技术之间关系的问题。亚马逊事实就是一家技术公司，几乎没有外包的项目，除了物流这样的第三方服务之外，它在技术方面基本上都是自主研发的，亚马逊的仓储物流体系是用很少的人高效做的。而京东商城和号称中国亚马逊的当当网，说到底，就是卖货的B2C网站。如果京东商城以后说自己是一个做仓储平台或者物流平台的公司，也是一点也不奇怪的，而如果京东商城说自己是一家做云计算的公司，估计所有人都不会相信了。当然，我要表达的意思也不是说卖货的就比做云计算的低一等，投资人最看重的还是投入产出比，我想说的是：电子商务的本质究竟是什么？我在上一期专栏里说：电子商务网站的本质就是互联网，它是一个互联网公司，就是花钱砸域名，砸出口的，卖什么无所谓。李树斌说：“电子商务是用技术和数据驱动的零售体系”，首先它应该是零售体系，它的定语是用技术数据驱动，以前我们想过用产品驱动，用用户需求驱动，我个人认为电子商务应该用技术和数据驱动，因为零售的本质应该是低价、多选择和便利性，如果没有技术和数据驱动低价根本做不到。多选择，一个正常百货商场里面很少有超过1万种商品，但亚马逊里有上百万种商品，只有技术的极大提高才可以应付这样多的商品。便利性就更不用说了，如果没有技术不可能实现。当然，我们是从完全不同的角度去定义电子商务的。我是从互联网、流量和入口的角度去谈，李树斌是从做贸易的角度去谈。

从做贸易的角度去讲，我同意李树斌这个结论：电子商务应该用技术和数据驱动，如果没有技术和数据驱动低价根本做不到。所以这也是为什么在淘宝为卖家服务的平台上，基于数据的量子统计和数据魔方卖得最火，而且很贵，卖家也愿意掏钱。再次，我强烈呼吁做电子商务的公司，加强在技术方面的投资，做一个技术和数据驱动下的电子商务公司，而不是只知道线下备货，线上上聚划算，上淘宝“双十一”活动这样的公司。



邢波涛
北京新软孚信息技术有限公司技术负责人。关注SaaS管理软件和B2B、B2C电子商务的融合。

通用商业语言 UBL

UBL可帮助商业需求被精确描述，减少标准重复开发并保证标准的延续性。

近日，OASIS发布了通用商业语言UBL（Universal Business Language）的2.1版本征求意见稿，这个版本在保持与UBL 2.0版本良好兼容性的同时，面向电子业务应用对UBL语言进行了多处增补。值得注意的是，在新版本中，UBL还针对财经业务进行了优化和增强，改进了UBL与财务系统基础设施的接口互操作性，并在标准UBL语言基础上添加了拓展，以实现借助XAdES格式对UBL文档进行XML高级数字签名的功能。

自2004年由OASIS技术委员会制订第一个版本以来，通用商业语言UBL就一直致力于促进电子数据格式的标准化，来保证企业间信息与数据交换的有效性和一致性。在UBL规范中，以通用XML库的形式给出了电子业务文档格式的定

义，包括产品预售、业务订单、数据交换、发票单据、付款支付等各个业务环节。在设计上，UBL遵循面向对象理念，能够在不同业务领域间有效进行条目重用。UBL具备严格的标签命名方式，设计者同时借助了CCTS语义原子性方法来保证UBL规范对信息描述的精确性。此外，在可拓展性方面，UBL文档的参考引用、标识符、代码段可以根据具体的业务环境进行定制和修改，并能够借助XML映射将UBL文档转换成各类其他文档类型。

对于特定的大型行业信息化系统建设来说，准确清晰的业务逻辑描述往往比具体功能的技术实现更为关键。UBL提供了将电子业务交易过程中纸质文件转换为以XML格式为载体的非结构化信息标准文本格式的方法和途径，借助通用的标准规范，来保证企业在进行电子业务在线交易时，使用标准通用的格式进行文档和数据交换。

UBL规范吸纳了业界多个数据标准化组织共同参与，以便于UBL能够直接植入已有的在线交易、业务审计、记录管理等业务系统中使用。此外，UBL自2.0版本以来，就与电子数据交换格式EDI（Electronic Data Interchange）及其派生的XML标准保持兼容，以便融合EDI中定义的贸易、运输、保险、银行、海关等行业信息交换格式，在以贸易为中心的电子业务应用系统中进行处理。

创建于1993年的结构化信息标准促进组织OASIS一直推进着UBL规范的发展和更新。OASIS最初以标准广义标记语言SGML（Standard Generalized Markup Language）组织的形式成立，旨在推动产品互操作性架构的建立以及SGML语言的定义。在1998年更名为OASIS开放组织后，将业务拓展到更为广阔的范畴，并成为推动电子商务标准发展、融合及采纳的非营利性国际组织，为数据安全、网络服务、接口一致性、电子交易、产品供应链、公众服务以及企业间互操作提供通用性标准。OASIS制订的各种规范帮助商业需求被精确描述，减少标准重复开发并保证标准的延续性，在协调各行业标准化组织的基础上通过一系列基础性工作推进软件供应商之间的合作。更为重要的是，OASIS各项标准均以免费且公开的形式向公众开放使用。

目前UBL已经在欧洲政府采购联盟PEPPOL（含12个欧盟成员国）、电子货运管理（EFM）等多个领域内得到成功应用，如果你所在的机构感兴趣使用UBL定义的低成本、迅捷、安全的电子文档处理方式来提高工作效率，升级传统业务流程，不妨访问UML官方站点（ubl.xml.org），更进一步了解UBL通用智能语言规范的细节内容。



高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

丑小鸭的重生

Lumia 800只要不出现大的质量问题、断货问题，注定会成为这个圣诞节的主角之一。

10月26日，诺基亚在伦敦举行的Nokia World上正式发布了第一代基于Windows Phone 7的智能手机——Lumia 800和710，距离2月22日宣布支持Windows Phone 7平台，正好八个月的时间。八个月前，Google的高管曾经说过，两只火鸡变不成鹰。但是，小天鹅在刚出世时也会被人当做丑小鸭。

首先来看Lumia 800，这款被太多人寄予厚望的机器，采用了AMOLED ClearBlack弧面玻璃屏幕，配合三种颜色的外壳，给人一种耳目一新的感觉。我实测过HTC莫扎特、三星Focus和Lumia 800，在屏幕的色彩还原度和饱和度上，Lumia 800完胜，在强烈阳光下，Lumia 800特有的屏幕仍然能够清晰显示。虽然没有采用Cortex A9的双核CPU，但Lumia 800的整体性能还是得到了一致的认可，“流畅”成为所有使用者的一致评价。

在软件方面，Lumia 800内置了Nokia驾车等特有应用，这也说明Ovi Map在美国之外的地区，的确超过Bing Map，这也会成为Nokia手机进入中国时的撒手锏。

在Lumia 800的研发上，Nokia显示出一贯的稳健作风，在N9的基础上对于硬件进行了部分改造，相同的CPU也使得移植工作变得简单。这也是为什么Nokia能够在短短八个月的时间就可以推出一款量产手机的原因。Nokia在全面转向Windows Phone 7后，仍然推出了一款MeeGo手机，也为自己留下了一颗种子。不过在未来的几年中，Windows Phone仍旧会成为Nokia唯一的选择。

当初在Nokia选择Windows Phone时，很多人看不懂，其实Nokia的选择很容易解释，当延续十多年的Symbian在硬件快速升级的时代不堪重负，而被寄予厚望的MeeGo迟迟不能成熟的几年间，Nokia错过了在Android阵营中取得主导权的机会，Nokia不可能在昔日对手MoTo或三星的身后当小跟班，所以，Windows Phone成了Nokia唯一的选择。

当Google收购MOTO之后，微软又变成了唯一不生产硬件的移动操作系统提供商，虽然之前很多人呼吁微软收购Nokia，但为了拉拢HTC、三星等厂商，微软恐怕不会采取贸然的行动。三星在Android、Windows Phone和bada之间消耗了太多的精力和时间，很有可能像两年前的MOTO一样，在多个操作系统间游移，反而错过了下一波技术浪潮。HTC是以Windows Mobile起家，Windows Phone的移植也是由HTC最先实现的。但HTC已经在Android上倾注了太多精力，这就注定了HTC在Windows Phone领域是一个跟随者，而绝非领导者。

Lumia 800只要不出现大的质量问题、断货问题，注定会成为这个圣诞节的主角之一。比起黑白两色的iPhone，色彩鲜艳的Lumia 800一定会吸引很多时尚青年，尤其是青年女性的注意力。而Lumia 710则是主打性价比，相比高端Android手机，便宜、操作流畅都会成为Lumia 710的卖点。Nokia一直都是一家快销公司，研究用户心理、制订价格策略方面，拉高冲低的手段早已玩得驾轻就熟了。

Windows Phone的竞争对手怎么办？iPhone推出更加物美价廉的减配机型，推出彩壳版iPhone，压缩Lumia 800的价格空间。虽然会损失一些盈利空间，但比起Lumia 800来，iPhone更加成熟、拥有更多应用，所以对于最终用户来说，仍旧是首选。Android的情况复杂一些，Google在收购MOTO之后，如果不能通过Android 4.0切入高端市场的话，也会影响Android这个松散的联盟在中低端手机市场上的影响力，很多缺乏自主研发能力的OEM厂商正在观望，一旦Windows Phone获得成功，OEM厂商们阵前倒戈的可能性也不是没有。

好戏开场，接下来的一幕是关于两只火鸡如何变成鹰的故事。



马宁
北京支点联游CTO，OpenXLive开发者。曾任微软亚洲工程院软件工程师，从事Windows Embedded Compact开发工作。加入微软之前，连续四年获得“微软最有价值专家”称号。

一个更大的筐——SQL Server 2012

SQL Server 2012这个筐对于大多数用户而言已经够大了，但功能特性的组织上似乎还欠点章法。

在刚刚举行的SQL Server专家协会峰会上（PASS Summit：The Professional Association for SQL Server Summit），一个最值得关注的主角缺席了——SQL Server 2012，而且预计要到明年上半年才能露面。

对于很多SQL Server的DBA们而言，面对SQL Server 2008 R2，也许偶尔会感觉自己Out了——除了以往传统意义上的“数据库”以外，SQL Server 2008 R2就好像一个大筐，充斥着各种新技术、新工具。不仅如此，以前DBA们只要管理企业自己的数据库，现在还要加上一个“云”里面的数据库。不过，就好像微软的其他产品一样，SQL Server 2008 R2这个筐里的很多“新鲜特性”只是“小白鼠”，真正让他们“堪大用”恐怕要等SQL Server 2012再说，原因很简单——用户在试用、扩展这些功能的同时，微软也在做类似的工作，毕竟“新特性”真正用于企业环境需要等待开发、系统管理、运维保障、信息安全等配套手段基本就绪才可以，而SQL Server 2012这个更大的“筐”里就装了不少SQL Server 2008 R2中“新特性”所欠缺的这些手段。

此外，之所以称SQL Server 2012为“筐”，还因为很难从官方的介绍材料中把这些新、老功能组合在一起的脉络理清，尽管我们可以笼统将他们归纳为“更强大的系统管理功能”、“更强大的BI功能”和“完善的云信息服务功能”，但怎么看它都不像是一个单纯的数据库产品，而是一个服务整个企业“当下”信息管理的平台。

■ 尽管在各种做数据库同行的聚会中，大家或多或少会把自己企业管理的数据量、增长量作为衡量彼此数据“家底”的标准，但站在我们的用户角度看，“海量”一词听得太多，就好像“虱子多了不痒”一样，已经不

是他们关心的问题，最起码不是他们眼前棘手的问题，不过“数据质量”确实是摆在他们面前必须实实在在尽快解决的问题。因此，SQL Server 2012中覆盖全数据生命周期管理的主数据、元数据、数据质量评价、“查重”、摸底“数据血缘关系”等一批管理功能的完善到很值得期待。

■ 其次就是“云”。对于企业而言，没有自己的数据，那么上层的各种所谓“云服务”只是浮云，下层的各种“云基础设施”也只是水蒸气，和自己关系不大，但SQL Server 2012则是在微软认真基于云平台开发了整整一代产品之后浓缩而成，它把信息管理的主动权交给用户，由用户自己根据企业内不同业务流程、不同数据需要在“公有云”、“私有云”甚至传统数据库存储的“自留地”里管理自己的数据，所有的工作都只在同一个SQL Server管理环境完成，而不需像现在SQL Server 2008 R2那样一会天上一会儿地下的折腾。

■ 在商务智能分析方面，尽管底层技术算法变化不大，但从便利用户商业使用看确实做了不少“实惠”的功能，购物篮分析、网站分析、预测、无监督学习、非结构化数据探索性分析，这些功能提供得恰逢其时。毕竟面临市场环境中众多的不确定性，用户应该将精力放在采样、分析、发掘自己关心的商业数据上，而不是花了大把时间和钞票去完整学习SQL Server分析算法的使用上。从发布的特性看，最常用的分析情景SQL Server 2012都提前给您“备”着了。对于规模更大的企业，SQL Server 2012甚至可以作为经营策略的“数据实验室”，结合VSTS（Visual Sutdio Team System，微软的开发平台）的自动化测试工具，可以在经营策略具体实施前，先在SQL Server 2012中多次“预演”，然后再通过相关的分析、预测功能做实验性检验。

■ 开放性方面，SQL Server 2012也有不少进步。比如：对于XML标准和采用XML描述的各种数据格式的支持、对于Apache Hadoop的支持。毕竟更加“包容”才能愈发显示出这款产品的“大气”。

仅从功能看，SQL Server 2012这个筐对于大多数用户而言已经够大了，但功能特性的组织上似乎还欠点章法。也许在后面半年左右的时间里，微软能通过最佳实践、指南等方式帮助用户梳理他们，而不仅只是针对一个个功能点发布一批HOL（动手实验：Hands On Lab）。



王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

业界报道

Google开发者大会在京召开



Google中国工程研究总经理杨文洛

10月27日，主题为“开放•分享•创新”的2011 Google开发者日技术大会在北京成功举行。来自Google的工程师和技术人员与开发者们，分享了Google的创新技术对于全球及中国开发者的重要意义，以及对于产业和社会所产生的影响。

参与本次GDD大会的嘉宾包括Google中国工程研究总经理杨文络、Google技术推广首席专家Michael Jones、Android全球开发者生态系统总监Eric Chu、Chrome产品团队的高级产品经理Jan Linden。

本次大会分享了大量的Google产品数据：现在每天有超过50万台Android设备被激活，每天超过10亿用户使用Google，每天超过1亿用户使用Google Maps，拥有超过13亿的Gmail用户，YouTube视频每天被点阅20亿次。

Google还向大家展示了代号为冰淇淋的Android 4.0系统，在中国有超过600种Android终端，价格覆盖高中低市场，这一全面改进的智能手机系统将会继续改变大家的生活。

本次大会设置了 Android平台、Chrome浏览器、HTML5&CSS3、Google Maps技术、AdMob、创业融资等多个种类的开发者技术讲座，与开发者分享开发产品和相关技术的最新发展，展示最新互联网技术的应用、探讨如何利用这些新型的开发技术来推动创新和创业。此外，2011 Google开发者日技术大会于11月4日和11月12日分别在上海和广州举行，与更多的开发者分享Google开发产品和相关技术的最新发展。

一位内核开发者的见闻

记Linux Kernel Summit 2011大会

作者 / 韩莹

Linux Kernel峰会（简称KS）是Linux最重要的年度大会，受邀的是开源组织各个子系统的维护者和核心开发者。今年的峰会安排在10月23-25日。作为Google内核开发组和Linux开源开发的一员，本文作者受邀参加了今年的KS大会。文中记录了一些印象较深的讨论。

What's Next For Control Cgroup

Cgroup是内核里用来把用户进程分组的机制。在此基础上每个子系统（CPU、Memory、Disk和Networking）有相应的机制来监控和限制资源利用。用cgroup和resource controller来实现多个任务的资源共享，同时提供每个任务运行环境的隔离，从而保证任务性能的稳定性。由于与现有的Virtual Machine在系统性能上保有优势，包括RedHat、openSUSE、Google、IBM、Oracle和Parallel等在内的公司都在一定程度上采用了cgroup。

由于越来越多的用户需求，今年KS上专门安排了有关cgroup的讨论。James Bottomley首先提出了目前cgroup开发社区资源不足的问题，包括开发人员和维护者。现有维护者由于某些原因即将退出，大家一致认为需要马上找到新的维护者。Andrew Morton指出很多在内存管理社区的patch都是和cgroup相关的，现在的问题是没有专门的cgroup开发人员参与讨论和做Code Review，相应的patch进度就会被放慢。当然，同样的问题在其他子系统里也会出现。James在会后创建了一个新的邮件列表（cgroups@vger.kernel.org），除了针对cgroup的讨论外，所有子系统的controller的讨论也建议抄送到这个列表上。不管cgroup现有的实现是否理想，但用途已经越来越广。包括Andrew和Linus都在会上提到cgroup的发展是不可逆转的，接下来也希望多一些社区开发者加入到其中的讨论中。

Memory Controller（memcg） Workshop

Memcg建立在cgroup的基础上，支持memory isolation机制。这次难得的机会，很多核心开发人员包括维护者都来到了布拉格，所以在LinuxCon会议期间我们组织了个专门的Workshop。包括来自Google、RedHat、openSUSE、Fujitsu和OpenVZ的十几位工程师在一起讨论了当前的开发进度和之后的开发计划。这次的Workshop我们主要针对最近的几个开发项目进行了讨论，这里简单介绍一下几个重点项目。

- Kernel Memory Accounting：目前3.1内核中memcg只支持user page accounting，但由于用户进程也会申请大量kernel memory，没有相关的kernel memory accounting会严重影响程序运行性能的稳定性。Google和OpenVZ都在参与相应的开发，目前主要的挑战是怎样在大规模网络服务器的环境下运行并不引入系统regression。
- Soft_limit Reclaim In Over-committed Environment：Over-commitment是普遍采用的提高系统资源利用率的配置。在今年的LSF上我提出利用memcg已有的Soft_limit接口在page reclaim里实现over-commit得到了积极的认可。这次讨论包括RedHat、Google和openSUSE在内的工程师把现有的实现做了进一步改善，之后会有具体细节发布在linux－mm的邮件列表里面。
- Per-memcg Dirty Limit Accounting：Linux支持设置系统允许的dirty page数目的上界，但没有支持针对每个Cgroup的支持。如果只是依靠系统的设置，很容易造成Cgroup被Out-Of-Memory Kill。Google的实现方法得到了大家一致的认可，接下来应该很快被引入upstream里。

这次会后一个很大的感受是越来越多核心内存管理的开发者加入了memcg的讨论和研发。记得2010年10月我去渥太华参加Linux Symposium（OLS），当时和IBM的Balbir Singh（memcg的作者和维护者）讨论接下来的开发项目和相关细节，大部分的讨论都是我和他在会后进行的。今年在旧金山的Linux Storage and Filesystem Summit上，很大一部分会上时间就开始讨论memcg相关的开发细节了。所有这些变化很



开源组织各个子系统的维护者和核心开发者济济一堂

大的一个推动力是不断增大的用户需求，Google的云计算平台和OpenVZ的虚拟计算平台都是基于Container的，RedHat和openSUSE近期的distribution都有cgroup的支持。和Mel Gorman会后谈到这个问题，统计过去一段时间内存管理的patch，大部分都是和memcg相关的。同时我们都希望能有更多的工程师，尤其是核心的内存管理和文件系统的开发者加入其中的讨论并给出修改意见。

What to do with Android

今年的话题是怎样提高Code Review的质量。大部分子系统的维护者都谈了自己的想法，问题还是集中在没有足够多的资源和时间对每个patch做细致的Review。

简单介绍一下Linux社区开发的流程：所有的patch都要发布在lkml的邮件列表上。作者的名字需要标注在“Signed-off-by”后，当然一个patch也可能有多个作者，最后一个修改过的“Signed-off-by”出现在最下方。比如所有的内存管理patch都要先进入Andrew的mmotm tree，然后Linus会pull mmotm，所以大部分的patch最后两个“Signed-off-by”是Andrew Morton和Linus Torvalds。一个patch一般都是要经过几轮的code review，最终才有希望被接受。当然也有些patch一开始就被否定掉了，主要原因是要解决的问题没有得到一致的认可。如果被多个人Review过，每个Review的人会在Email里打上“Reviewed-by”（相对仔细看过patch的细节）或是“Acked-by”的标注（大致看过patch的细节）。之后每个子系统的维护者会根据情况把patch加到各自的tree里，最后由Linus决定把各个子系统merge到Linux的tree里。

其中最关键的一步是把要解决的问题阐述清楚，不要一开始就关注实现细节。建议如果是大的feature proposal，最好先发布一个RFC然后附有patch的prototype。另一点是要把正确的人抄送进来，一般是包括维护者和最近在相关代码中改动很多的开发人员。建议多花一些时间做测试，一定量测试结果会大大增加reviewer的信心。最后也是最关键的一点，是要得到用户的认可和支持。每个feature的开发目的都是要解决一个实际问题，就像Linus在会上对Android的评价：“Code that actually is used that is actually worth something。”

最后给工程师一点建议

贡献patch到开源社区一般需要花几倍于开发patch本身的时间，但受益是长远的。有的patch最终没有被接受，有时只是时间的问题。最难的环节一般在开始，怎样解释清楚要解决的问题。如果问题本身被接受了，接下来的实现方法就容易很多了。最后提一下测试程序，这个十分关键，我们很多时候花太多时间去描述要解决的问题，但一张测试结果图通常胜过千言万语。

“用户体验”以人为本

记UserFriendly2011暨UPA中国第八届用户体验年会

记者 / 陈秋歌

用户体验影响着产品的成败。提高产品用户体验成为企业高管及产品研发人员关注和探讨的话题。2011年11月11-13日，以“设计拥抱变革”为主题的UserFriendly2011暨UPA中国第八届用户体验年会，在苏州科技文化艺术中心召开。

本次大会上来自美国、德国、荷兰、瑞典、韩国、中国及中国香港、中国台湾等国家和地区的40多位行业专家与前沿设计师，给近800名参会人员带来不同领域的专业分享。

用户体验越来越受到企业和用户重视，成为当前时髦名词之一，作为从事用户体验的管理者及专业人员，要从光环背后看到当前用户体验所面临的诸多挑战。

用户体验即快乐的感觉

作为主管联想创新设计与用户体验的副总裁，**姚映佳**表示当前用户体验仍处在探索阶段，还没有明确定义。他认为成功的用户体验可以给用户带来积极的“感觉”，这种感觉包含满足与快乐。

如何达到这种快乐的效果呢？**姚映佳**列举出可以通过增加高品质的细节、创新的产品形态、及时的惊喜体验、将无聊的体验和任务与环境产生更多互动等方式来增加用户使用产品的快乐感受。

合理的组织可以帮助用户体验人员发挥自身最大价值，如何构建设计和体验组织呢？现在基本上有两大组织方法：Top Down可以完成一个自上而下、从高层到底层的全局方向和战略上的把控，有利于任务高效完成，但会造成与事业部沟通的疏远，存在较大风险。而Bottom Up很容易带动各个事业部间的互动，但在设计的一致性和持续性上存在一定的问题。**姚映佳**表示，联想经过摸索总结出了混合型组织架构，将Bottom Up和Top Down融合在了一起，继承了两者的优点。

姚映佳强调，用户体验人员应该将更多的视线投向社会，从社会的角度去看待体验时，可能会获得超出想象的收获。

以人为本，更好融入社交网络

社交网络的影响力正呈现一种爆炸式增长，而人是社交网络的中心，这也成为行业专家探讨的重点。

荷兰Delft理工大学工业设计工程学系教授及系主任Pieter Jan Stappers认为深刻理解人们生活的细节，通过分析他们朴实的自然需求而形成对用户深刻、细致的理解是产品设计最重要的信息渠道和资源。

与用户沟通时，他们提供的往往是表面显现的信息，深入挖掘隐藏在背后的真实信息才是真正考验设计人员的地方。挖掘只能通过长期不断的研究来实现。Pieter Jan Stappers指出有两条原则可以遵循，一要逐步地了解用户，不要直接笼统地询问他们，可以从具体事件入手，逐步挖掘更多信息；二要寻找合适的沟通方法，让用户自然地把情感和想法表达出来。

Facebook全球品牌体验经理Paul Adams认为，人正成为互联网的中心，而设计要以人为本。在互联网以人为中心进行重建的过程中，企业也要围绕“以人为本”重新建设，原

因很简单，如果你不这样做，你的竞争者肯定会这样做。

以人为本，了解用户的动机，首先要明确人生活在社交网络中。大多数人都有4~6个朋友圈子，而朋友对朋友，又将辐射出一个更为广阔的社交网络。设想一下，你的产品直接影响到其中一人，这个人又会将该产品的影响力推广到多大范围呢？1千人、1万人，甚至更多。根据调查研究，现在名人效应对用户购买行为的影响越来越小，而亲朋好友的建议成为用户购买的主要参考。

三星电子高级副总裁**张东薰**分享了用户体验发生的改变，如智能手机与SNS的结合，为人们提供了更广阔的交流平台。SNS与位置、游戏、音乐等各种媒介的结合，又为人们创造出前所未有的独特体验；云服务的推广，可以实现信息实时共享与传输，以及手机、电脑、电视间的互通互联；智能设备与NFC的结合，使人们的购物、旅行变得更加便捷；增强现实技术实现了现实世界与虚拟世界的无缝整合。

确定用户群对产品设计至关重要

在“移动改变世界”为主题的移动互联网专家论坛中，专家分析称，社交化将成为未来产品的一大发展方向。小米科技联合创始人**黎万强**指出，SNS产生之前，为产品寻找营销渠道是最痛苦的事，而SNS的盛行，利用类似微博、人人网这样的平台，信息到达通道扁平化，好产品被瞬间放大，为营销带来了变革。

创新工场首席布道师、软件架构师**蔡学镛**认为好产品及其产生的好用户体验，背后都需要技术来支撑。他建议从事用户体验的人在未来可以了解一些技术，这样有利于设计出更好用户体验的产品。

果断判断产品的用户群，是设计一款产品的基础。具体如何展开，还是一个难点，小米科技联合创始人黎万强认为。同时新浪无线事业部用户设计部主管**陈霖**也表示用户需求的过滤及过滤后的实现，是产品设计过程中遇到的难点。

最后Facebook全球品牌体验经理Paul Adams表示，虽然大家都在讨论以移动优先，但不必要局限在移动这一领域。为人们设计出好的产品，造福于全球几十亿的用户才是成功的关键。

沟通、协作能力是做好用户体验工作的基础

Albridge Solutions用户体验设计副总裁Cindy Lu 、SAP用户体验副总裁Andreas Hauser、唐硕咨询首席咨询师**黄峰**、华为UCD与全球化部部长**王菁**和青蛙设计执行创意总监Rainer Wessle五位嘉宾在主题为“好的管理成就好的设计”的设计管理专家论坛中，就如何提升用户体验对公司的影响力及设计管理流程等问题进行了探讨。



人本、变革、协作成为本次大会的主题

Rainer Wessle认为要想使交互设计团队在公司中脱颖而出，首先，要注意陈述的对象和讲故事的方式。对于公司高层，如CEO，他们最关心的是我们究竟能创造出怎样的成果。这时我们要抛开厚厚的纸制产品规格文案，通过合适的表达方式，把产品方案通过讲故事的方式告诉高层，让他们有一个直观深刻的理解。其次，交互设计团队要从设计过程中遇到的各种各样的局限中提出解决方案，这才是交互设计师的意义所在。

Andreas Hauser认为，用户体验管理者首先要真正了解用户体验团队及该团队可以带来的价值。既能帮助企业制定未来愿景，又可以为此执行细节设计出成功的产品，是用户体验团队未来的发展方向。Cindy Lu补充说，用户体验管理者要思考用户体验团队的说服力在哪里，想办法用对方的语言说服对方，让他意识到他可以帮助你、帮助公司、帮助完成他所想要的东西。

用户体验人员的实践工场

除了精彩的主题演讲外，大会还安排了近30场工作坊。工作坊不同于主题演讲，来自一线的用户体验专业人员带领参会者参与到具体项目中，其操作性、趣味性更强。

工作坊内容涉及用户体验技术的方方面面，如设计原则、设计方法、敏捷设计、开发工具、用户研究、典型案例分析等，可谓用户体验专业人员的一场技术盛宴。

2011年 Ruby技术大会：松本行弘谈如何选择开发语言

记者 / 夏梦竹



松本行弘建议中国程序员一定要有属于自己的思想

一年一度的中国Ruby技术大会11月11日在上海光大会展中心成功落幕，本次大会由CSDN主办，ITeye、Shanghaionrails协办，当天有超过500名来自全国各地的Ruby技术爱好者亲临现场。作为近年来在Web开发方面效率很高的语言，Ruby正引起全世界技术者的关注，应用范围也已扩展到企业领域，众多移动应用和云计算开发都在考虑采用Ruby。

Ruby语言创建者、Heroku首席架构师松本行弘发表了主题为《One Size Does Not Fit All》的演讲。他强调，没有万能的单一解决方案，但Ruby提供了多种选择。在介绍了JRuby、Rubinius、 MagLev等实现后，他宣布了针对移动开发的mRuby。他多次建议大家将Ruby迁移到1.9版本，并为大家详细讲解了1.9版本的新特性。1.9版本会对1.8版本保持100%的兼容性，代码也会更简洁、更有效率。会上他透露Ruby 2.0将于20岁生日的时候发布。

松本行弘在现场接受了记者采访，在谈到开发者该如何选择开发语言时，他认为，所有语言都遵循图灵等价原则，在理论上功能是等价的，然而不同的情景模式下有着不同的开发成本，面对目前越来越快的硬件支撑，更高的开发效率与易于理解的代码成为开发者的主要选择之一，而Ruby正符合这一理念。对于国内广大外包项目程序员，他的建议是一定切记要自主性、决定权和属于自己的思考。

此外，松本行弘分享了一个有趣的观点：相对于数学家来说，程序员更需要倾听、理解和沟通，因为编程人员更需要揣摩使用者的感受和真实需求。松本行弘说他小时候数学很差，英文和语文很好，又特别喜爱计算机，他认为在编程的广阔领域中，虽然也有不擅长数学就什么都做不了的内容，但大部分是与数学没有什么关系的，包括Ruby的设计过程，大部分都没用到数学。

Intridea公司合伙人、Ruby社区著名开源项目OmniAuth的作者Michael Bleigh从过程、声誉、业务分享了他的程序人生，并给出了职业建议。随后国内资深Ruby实践者轮番登场，VMware中国软件工程师王浩飞介绍了Cloud Foudry，作为VMware用Ruby实现的首个开源PaaS平台，支持多种语言以及开发框架，能集成各种后端Serivce（MySQL、MongoDB和Redis等）支持多种云计算平台，能提升开发者的生产力，加速软件开发部署。

下午的主题演讲 “ 干货 ” 尤其多，ITeye（JavaEye）网站创始人、CSDN网站产品总监范凯谈了Rails3.1的特色和开发实践心得，ITeye网资深会员莫枢的 《Coroutine与Ruby》，淘宝广告业务部李建业的《运维系统开发与Rails 3页面开发实践》，淘宝测试部黄小勇的《谈rails3在淘宝测试平台的应用》，淘宝MED部李华顺的《Rails项目运用Redis构建高效的实时搜索功能》，Intridea公司亚太总监俞玶的《Making money with OSS》都获得了与会者的强烈反响。

本次上海的活动为期1天，安排在周五，却有全国各地的朋友赶来参加，早上3场，下午6场，满负荷的演讲，几乎无人早退。Ruby开发者的热情，Ruby社区的凝聚力，非常振奋人心。Ruby中国大会已经连续办了三年了，一届比一届人气旺，作为一门开发效率很高的语言，Ruby受到更多的开发者关注。



智能电视引发技术新浪潮

记者 / 陈秋歌

越来越多的开发者投身移动应用开发，移动领域竞争日趋白热化。从数以万计的应用中脱颖而出，已变得相当困难。在寻求另一片发展天地的过程中，越来越多的互联网公司和电视厂商开始把目光投向智能电视。智能电视或将成为开发者的下一片热土。

智能电视领域机遇与困难并存

电视是客厅长期的主宰者，存在着惊人的影响力和潜在盈利空间。智能电视作为一个开放平台，用户可以如同使用智能手机一样随意安装软件、游戏。它颠覆了传统电视单向广播式的观看模式，更具娱乐性和互动性。据中国智能终端多媒体技术联盟执委会委员梁铁航分析，智能电视正以飞快的速度发展，到2012年，在互联网电视中的覆盖率将达到70%以上。

智能电视的发展前景是美好的，而当前阻碍其迅速发展的却是标准不统一。各大电视厂商都有自己的一套标准，导致开发出的应用程序仅限本平台使用，这不仅是一种极大浪费，还严重影响了开发者的积极性。现在各大电视厂商都渴望能有一套统一标准，打破当前各自为营的发展僵局。2011年8月，中智盟的成立，为统一标准的建立带来了一丝希望。

中智盟，全称为中国智能多媒体终端技术联盟，是由TCL、长虹和海信自发组织的技术共同体，致力于为中国智能多媒体终端产业构建统一的行业技术标准。在中智盟成立之际，发布了首个应用程序商店技术标准SDK。第三方开发者基于该SDK开发的应用程序，可顺畅运行于三大品牌的智能电视上。

众多开发者开始投入智能电视领域

开发者不活跃的平台，不能算作成功的平台。长期以来，由于没有良好的商业模式，投入该领域的开发者并不多。近几年，随着智能电视逐渐被关注，很多企业开始专注于该领域，并推出了一系列创新型产品，同时也吸引了众多移动领域的开发者加入。

除了上文提到的标准问题，需借助于遥控器或其他一些遥控设备进行操作，这在很大程度上影响了智能电视的用户体验。梁铁航在接受采访时表示，在未来，用户手势识别、语音识别将会对智能电视的用户体验带来革命性的提升。投入智能电视开发的开发者，可以多关注这些方面的技术进展，以寻找产品技术创新点。盈利方式是每个开发者考量新平台的重要标准。应用商店的推出，铸造了移动应用生态系统的成功。智能电视同样将延用这一商业模式。“目前智能电视领域存在六大盈利模式，即互联网智能电视软件集成运营商、增值电信业务经营、电子商务、广告、网络文化经营及应用程序商店，中智盟相应地都提供了技术平台做支撑。比如在应用程序商店商业模式中，我们为广告系统做了标准的App接口，开发者只需要调用此接口，无需关心广告的内容及费用核算方式，即可直接对广告收入进行分账。”

结语

智能电视是继互联网电视后的又一个重要里程碑，它允许用户根据自己的意愿享受各种服务，而不必再局限于电视供应商提供的内容，这是一个巨大的变化。梁铁航认为智能电视操作系统的出现，将对智能电视的发展乃至整个电视产业带来巨大的发展机会，中智盟正在致力于智能电视操作系统的研究。

乐业背后的玄机

走进CA Technologies中国技术中心

记者 / 董世晓

前段时间，和一位在CA Technologies中国技术中心（CTC）工作八年的朋友聊天中得知，CTC有很多“高龄”员工。高龄员工？正当我一脸困惑的时候，他解释说这里的“高龄”不是指年龄，而是在CTC的工龄，无论外界如何变化，这些“高龄”员工一直在CTC兢兢业业地工作和生活。

说实话，这些年，各跨国IT公司的在华研发中心确实在不断发展壮大，但不时也会爆出有员工集体出走、核心成员离职的消息。而CA Technologies早在2000年就在华设立了研发中心即CTC，这些年可以用沉默有余、外在表现力不足来形容，但何以能够吸引住如此众多的员工“死心塌地”地在这里乐业呢？近日记者就走进CTC，一探其中的玄机。

共同的价值观

走进CTC，一种家的感觉迎面而来——绿植环绕的完全开放式的办公环境、个性化布置的工位、宽敞的休息区、我至爱的尽情畅饮的饮料——这不活脱脱就是一个放大版的家吗？我好像找到了一点答案。

当然，这些都是表象，关于更深层次的解答，CTC业务运营部副总裁Michael Tseng首先给我一张便签纸，然后介绍说：“我们将CTC的价值观印到便签纸上，以让全体员工时刻铭记。它包括四个方面，即：相互尊重、持续创新和进取、让团队精神闪亮、培养客户忠诚。CTC的价值观既是全球的，也是中国的。”

相互尊重：CTC的员工在进行沟通时，要有正直而坦诚的积极心态。每周二、三、四的下午，从两点钟开始，每半个小时就是一个一对一交流的时间，Michael会与各个经理分别交流。这半个小时，可以谈工作也可以随意聊天，以建立相互的信任。Michael也会利用这个机会给他们一些反馈，比如与A合作的其他经理不方便当面给A讲的话，Michael可以讲给他听，这就对中国人的害羞个性进行了弥补。此外，Michael还会与基层员工保持交流，比如在CTC开始提供免费饮料时，碳酸类饮料和非碳酸类饮料的比例是1:1。有位员工就对Michael讲：“你有没有注意到大家喜欢非碳酸类饮料更多一些？下班时往往只剩下碳酸类饮料，其他的都没有了。”也正是通过这种及时的沟通，CTC对饮料的配比进行了调整，更符合实际的需求。



CTC业务运营部副总裁Michael Tseng也是CTC老员工

持续创新和进取：在中国，一种能够证明是否创新的方法就是专利。而中国人可能比较保守，担心失败，往往习惯于被动的接受，不喜欢主动创新，因此，在引导员工进行创新方面，CTC做了很多尝试。一方面，通过积极推进敏捷研发模

式，让员工认识到创新能够带来效率的极大提升；另一方面，通过设立奖励模式，让员工认识到创新能给个人带来收益，以此激励他们进行创新。

除了技术课程之外，CTC还会为员工安排社交技能课程，教他们怎样沟通、怎样解决冲突、如何进行时间管理等，以便帮助他们实现无障碍创新和进取。

让团队精神闪亮：每到夏天，CTC会组织全公司的人出去玩；每个季度会让经理组织部门活动以及与其他部门的联谊。

培养客户忠诚：CTC的研发与销售在一起办公，通过Pre-sale项目，选出10名工程师，让他们每周抽出10~20分钟时间，与销售团队沟通、帮他们做产品介绍等。这是对研发和销售双方的考试，因为工程师在开发之外，应该有一些时间去面对真正的客户需求。这是CTC做的更接近客户的一个环节，即从研发的角度来看怎样接近客户。

以人为本，贯穿始终

在贯彻共同的价值观时，Michael Tseng特别强调了以人为本对CTC的重要性。他饶有兴致地提到，CTC是唯一一家将研发中心始终选址在长安街上的公司——开始是东方广场，现在是SK大厦。

对于选择办公地址，Michael Tseng有两个原则：交通便利，便于员工上下班；研发一定要与销售在一起，才能保持信息及时、畅通。尽管不负责销售，但他每周都会参加销售经理的会议。通过这两个原则，CTC筛选出了四个候选地址。

而为了能够敲定最终的办公地址，CTC专门召开了一次全员大会。会上对四个候选地址进行了非常详细的介绍，包括优点、缺点、出地铁后走路到公司的距离甚至卫生间的尺寸。听完介绍后，由全体员工投票决定选择哪个地址。结果是没有一位员工由于办公地址的变迁而离职。

听到这里，我不由得感叹CTC这样细致的以人为本的做法，与老罗今年在保利影院演讲中提到的为学生卫生间安装门栓的做法如出一辙，都将体验做到了极致，确实很值得借鉴。

虽然对与公司有关的任何事务，领导层都有决策权，但如果在一些能够调动员工积极性、方便员工的地方，将决策权交给员工，势必会收到意想不到、事半功倍的效果。

前瞻性的技术导向

当然，如果只有上面谈到的这些，会让很多人认为CTC是一个不错的“养老院”，对个人的发展并不十分有利。但随后的交谈打消了这种顾虑。



以人为本，畅所欲言的交流

大家都知道云计算的一项核心技术就是存储和虚拟化，而CA Technologies早在云计算概念问世之前，就已经在做存储和虚拟化方面的工作，而且早已成为其核心竞争力。而CA Technologies的ARCserve系列储存产品线，在CTC成立之初就一直由国内来负责，80%的功能都由CTC开发。正因如此，CTC的工程师们在云计算方面的储备和经验，已经走在了世界的前面。

更为重要的是，CTC不是做简单的产品本地化工作，而是能够直接研发产品，全程参与到从研发、测试、本地化、各语言版本直至客户支持这个完整的流程。这便使得员工能够最大限度地学到东西。

在中国设技术中心，值！

近年来，中国的人力成本不断增长，与印度或者其他新兴市场相比，已经不再具有优势。这是否意味着在华的技术中心将进行转移呢?

对此，作为CTC主管产品开发的老大，Michael的看法是：“不管对哪家公司来说，人力成本都是比较关心的话题。总部对此的反应是‘你觉得CTC在中国值不值？’从我来CTC到现在，我们只用了五年的时间，就把整个产品线从美国搬到中国，已经不可能再拿到别的地方去，因为已经在中国生根。况且我们也培养了很多的人才出来，CTC还会继续在扩大。在任何地方都是这样的，你找到适合的人才会做出对的事。此外，人民币升值也是人力成本升高的其中一项因素，不过，在我看来，人民币升值不见得是坏事，因为我们销售的产品会折算成更多的美元。总之，我们觉得，CTC非常值得继续做下去。”

架构师挑战与武器系列

如何画架构蓝图

文 / 王小虎

上期IBM大中华区合作伙伴技术支持总经理王小虎的“如何界定项目边界，把握系统全局”文章发表后，“如何画架构蓝图”成为读者最期待的主题之一。本期《程序员》特邀请王小虎就该话题分享了他的见解。

很多架构师在做架构设计时，知道要画架构蓝图，但不清楚究竟该画什么样的蓝图？如何画？是否有一些通用的方法可以遵循？

画架构蓝图4个常见的误区

■ 不画架构蓝图，直接进入概要设计。

■ 无目的无针对性。比如某架构师在项目方案架构设计中，画了一个如下所示的开发层面的架构蓝图：



给公司领导看的是这幅图，给业务人员看同样是这幅图，给设计人员看还是这幅图，用客户的话来说，知道这幅图不合适，但说不上哪儿不好，不知道该怎样去改。

■ 不知道怎样画架构图合适，只能凭经验凭感觉去画，这是很多架构师的心声。有没有画架构蓝图的标准方法呢？现在的做法往往在一幅图中表现了太多的元素。其中有企业层面的，有业务层面的，还有运营层面的，比较难看懂，而且表述方法不一致，比如上图中的节点有的用方框表示，有的用机器图表示，让人很费解。

■ 有架构师建议不画架构蓝图了。但这样做的代价是显而易见的：第一，客户、用户不能及时了解到整体的结构和风格；第二，项目团队成员不能把握大局，问题不能在早期发现，以至于在能容易补救时没能补救，发现时补救代价却已太大；第三，用户和项目组成员不能就相关方面及时找到达成共识；第四，用户和项目组成员需要长时间学习和了解系统。

架构师武器

首先来看**架构蓝图的定义**，架构蓝图是IT或者企业层面的示意图，展示了主要概念元素及其之间的关系，这些元素包括子系统、组件、节点、连接、数据模型、用户和外部系统。

出于沟通目的，架构蓝图应力图简单、简洁和便于理解，避免复杂、精确的细节。

因此，视图一般会采用一些丰富的符号、画面和一些支持性文字，用以解释架构的主要概念。

架构蓝图的目的包括：

第一，供项目出资人和涉众之间沟通交流，建立IT应用的概念；

第二，为开发团队提供架构和IT应用范围的概要愿景；

第三，探究和评估架构的备选方案；

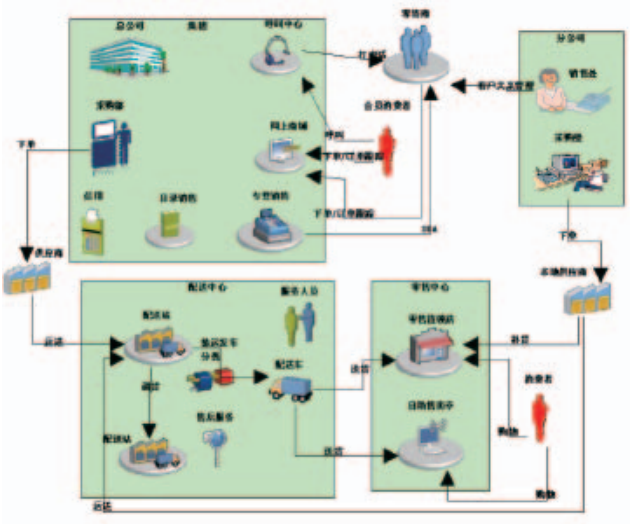
第四，尽早对架构方法的不确定因素达成共识；

第五，促进利益攸关者和开发人员之间的有效沟通；

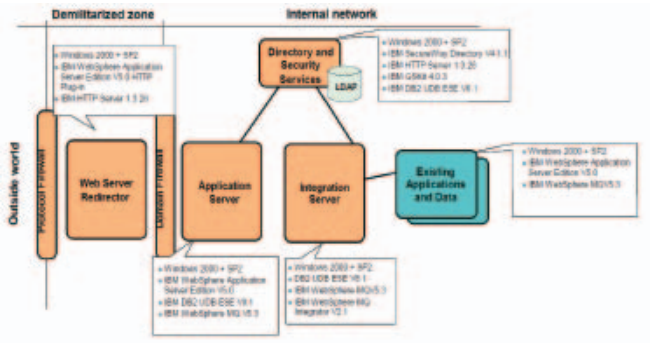
第六，促进刚加入项目的新人熟悉背景。

此外，用于企业级的架构整体视图还会帮助出资人和涉众之间就未来IT环境的全面发展方向达成共识，以此帮助管理层对IT战略投资和需投入的资源做出决策。架构整体视图提供了架构和IT应用范围的概要愿景。

架构蓝图可以大致分为两大类：企业（业务）层面和IT（技术）层面。一个典型企业层面的架构蓝图示例如下：



IT层面包括应用、数据以及技术架构，下图是映射了产品的架构蓝图示例：



在设计架构蓝图的时候有几点注意事项：

■ 首先，架构蓝图应能够适应各种备选架构的解决方案，能够供各涉众权衡。

■ 其次，处于IT方案级的架构蓝图应在项目前期产生（甚至早在项目立项前的方案建议阶段），并且将影响到后续的组件模型和运营模型。但这样做的目的并不是说架构蓝图要成为组件模型和运营模型开发和验证的必然依据。

■ 之后，随着组件模型和运营模型的定型，如果其主要概念和关系发生改变，架构蓝图也应做出相应调整。由于处于企业级的架构蓝图常常会成为整体IT战略的一部分，在这种情况下，架构蓝图应能描述企业组织所期望的业务和IT能力远景。

架构蓝图设计的方法

基本信息输入

创建架构蓝图通常需要包含以下8个方面的内容：

■ 业务远景和战略文档、IT远景和战略文档、IT能力、当前的架构远景材料和访谈，以及行业研究和技术概览；

■ 相关客户文档（包含了对项目目标的工作产品描述），推导出系统架构目标、约束和衡量成功的标准；

■ 当前IT环境的分析；

■ 通过“头脑风暴”方式产生架构远景视图的主要积木块；

■ 通过“头脑风暴”方式产生未来IT环境的主要特点；

■ 分析现有架构资产，包括参考架构和类似架构的描述；

■ 早期版本的功能性需求（通常以用例方式表达）；

■ 早期版本的非功能性需求和约束。

创建架构蓝图

首先架构师要识别出非功能性需求，因为它具有全局性的影响，并可能挑战用户满意度。其次应预见到系统未来可能的用户需求本质，以及什么样的系统才是适合的；此外，吸取以往的经验，找到系统概念的相似性，调查面向对象的行业模型及其他模型资产，套用架构模式也是极其重要的。

在分层的模式方面，可分为两部分：

纵向分层（Tier）——按照运营的物理场所展开；

横向分层（Layer）——在一个节点上（或多个节点抽象出来的一个节点代表上），按照依赖关系切分的层次。



套用架构模式，参考IBM红皮书，Applying Patterns Approaches Patterns for e-business Series；其他架构模式，比如Martin Fowler的企业应用架构模式。

在创建架构蓝图方面，还需要注意以下内容：第一，搜寻候选资产，评估需求和规格之间的差异，并评估差异对整体开发方法和工作量的影响；第二，设计决策必须整体上满足非功能性需求，这需要借鉴业界的经验和参考基准；第三，生成并评价技术选择方案，其中需要考虑行业的发展趋势；第四，通过原型（静态界面），流程图和丰富的图片，可视化展现备选的系统概念；第五，评估系统的实施可行性，如果有可能，构建风险识别和缓解的原型；第六，和项目发起人一起评估备选方案，并形成决定；第七，精细化架构整体视图，使其成为有效沟通的载体。

维护架构蓝图

由于架构蓝图是有效沟通和培训的载体，因此它必须与高层组件模型和运营模型一致。随着组件模型和运营模型草稿的不断变化完善，如果这些变化导致与架构蓝图不一致，要修订视图使其与其他模型、决策和原则一致。

参考资料：

IT systems Views_and_viewpoints什么是组件模型和运营模型。
IBM红皮书，Applying Patterns Approaches Patterns for e-business Series



王小虎

IBM软件集团大中华区合作伙伴技术支持总经理，中国IT行业架构方法论启蒙者和领导者，积极推动国内IT行业架构方法论的学习和实践，在培养国内IT企业的架构师方面作出了卓越的贡献。



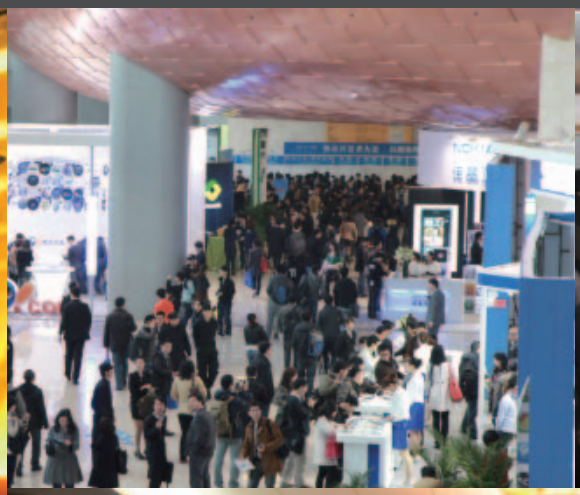
你准备好了吗？

2011中国 · 移动开发者大会侧记

记者 / 陈博

2011年11月3-4日，以“移动大变局下的抉择”为主题的移动开发者盛会——“2011中国·移动开发者大会”在北京国家会议中心隆重举行。此次大会由CSDN和创新工场主办，派代网、腾讯应用中心联合主办。两天内吸引了国内外移动领域知名的运营商、平台商、互联网企业、手机终端厂商、应用开发和投资公司的业界领袖，以及来自一线的研发、运营和管理的与会者超过5000人次，相信本次大会即是移动大时代的缩影。









黄金时代≠黄金时机

“这是一个最好的时代，也是一个最坏的时代”。

——狄更斯《双城记》

本次移动开发者大会开幕式的视频中呈现了这样一组数字：“2011年，iOS产品激活量已达到2.22亿；Android产品激活总量将达到3亿；移动终端销售量将达到18亿；苹果为iOS开发者分成已超过25亿美元；应用下载总量将膨胀至290亿次。”

同时，中国互联网络信息中心（CNNIC）在《第28次中国互联网络发展状况统计报告》中指出：“截至2011年6月底，我国手机网民规模为3.18亿，手机网民在总体网民中的比例达65.5%，成为中国网民的重要组成部分。”

基于以上因素，创新工场董事长兼首席执行官**李开复**用了“漫延”一词来描述今天的移动互联网。

设备的漫延：智能移动终端的大量普及；**用户的漫延**：移动互联网用户高速扩散；**娱乐的漫延**：移动娱乐平台崛起；**内容的漫延**：UGC内容主流化；**消费的漫延**：开始对用户消费行为全天候的影响；以及**创业者的漫延**：创业成本降低、天使投资蔚然成风。

与之相呼应的是会议现场的人潮涌动。据统计，本次大会的参会者中，除了移动应用开发者之外，还有大量来自互

联网与传统行业的从业人员。可见，大家都已经意识到，所有的IT应用都将移动化，所有的信息服务都将移动化。毫无疑问，移动互联网用户碎片化时间存在巨大的挖掘价值，加上创业难度的减小以及投资热潮兴起，使得开发者们拥有足够的理由确信已经迎来了史无前例的黄金时代。

但同时，CSDN创始人兼董事长**蒋涛**指出还有很多痛点困扰着开发者：**盈利问题**：目前真正赚到钱的开发者还是少数，本土用户不习惯花钱购买应用、App Store上遭遇黑卡问题；**来自平台方的竞争**：平台扩展导致应用封杀，团队在与巨头竞争时毫无优势；**运营成本凸显**：随着移动浪潮的大热，一款产品在亿万应用的海洋中仅靠口碑营销难以冒头；**人才竞争激烈**：行业优秀人才积淀不足，还面临着巨头挖角和员工辞职创业的风险；**创意雷同，山寨横行**：缺乏积累和创新，盲目跟从产品构想；**远期成长天花板**：创业团队自身计划不明确，只着眼于短期产品；**底子薄，成功难以持续**：一个团队也许能快速成功，但难以延续发展，完成从团队到企业的转型。

于是，“移动大变局下的抉择”成为了本次大会的主题。黄金时代已经到来，但对于每一位开发者来说，需要好好考量目前是否是自己投身于这股浪潮的最好时机，毕竟对于中小型开发者或开发团队来说，现金流无比可贵。

本次大会第一次听到来自官方的声音是主持人刘江的那句“你准备好了吗？”，相信这句话值得每一位开发者深思。

不仅仅是产品需要创新

“创新”是本次大会出现频率颇高的词汇。诺基亚大中国区总监**黄耀丰**表示，诺基亚抛开十多个亿的生意，穿起战袍重新向市场进军，就是为了更好的创新。微软全球资深副总裁**张亚勤**指出，本土开发者都想着创业，却忽视了创新。究其原因，金山软件董事长、小米科技创始人**雷军**给出了他的答案：“创新是件风险很高的事情，绝大部分的创新都会失败。对每一个创新者来说，最大的考验是有没有承受失败的勇气；对整个社会来说，是能否够宽容对待那些失败者。”因此，为失败者鼓掌，是整个社会对创新的贡献。

然而，即便创新者拥有巨大的勇气，社会给予他们足够的礼遇，山寨横行的市场却不断打击着创新者的热情。中国的开发者有一个很大的问题就是在盲从，开发门槛的降低也导致产品易于复制。所以一款产品在推出后不久便有众多类似产品蜂拥而至，虽然名字不同但功能上大同小异，尤其是几大巨头的“微创新”，往往使开发者无所适从。如果在经历了数次挫折后所创造的产品，却很容易遭到竞争对手的复制，那么开发者的热情将被消磨殆尽。

诚然，在面对山寨厂商和巨头平台围剿的时代，仅仅依靠产品内容的创新，最终只会被数以万计的复制品所淹没。所以，开发者应该意识到，创新不应局限于产品内容的创新，更应该包括推广与盈利模式的创新、对用户需求把握的创新、甚至于团队内部构建和工作流程的创新。一个好的创意赋予了产品生命力，但整个团队的持续成长往往需要持之以恒的创新动力，只有将创新深入骨髓，才能坦然应对多变的市场环境。

快时代让我们慢下来

顽石互动董事长兼CEO**吴刚**的演讲赢得了大家的赞誉。在这个普遍追求“快”的时代，吴刚为开发者分享了一些“慢”下来的理念：从单一平台起步，专注区域市场，针对细分市场的差异性逐步调整完善产品，保持团队自身的节奏感。

这种思路带来的结果，就是让产品精品化、用户群体精品化、团队精品化。很多团队希望能快速全面地覆盖市场，但最终的结果只能是不能左右兼顾，泡沫破灭。好的产品，周全的设计，完整的功能规划，其实不需要快速迭代。在大型公司中，暴雪是“托客”的著名代表，但其产品无一不是经典。小型团队里，《植物大战僵尸》的成功是极好的典范，其团队创始人John Vechey在自述文章中提到共花了3年时间来完善《植物大战僵尸》：“我们创业时，Facebook甚至还没有诞生，iPhone也不存在。我们适应、改变并且克服了我们所遭遇的一切。我们持续不断地将我们的方法整合到游戏中。我们永远做不到完美，但却一直都很优

秀，而且总是试图做得更好。”

移动时代人人都是创业者

在大会第二天，七大主题论坛异常火暴，产品与设计主题论坛尤为突出，不只是产品经理、设计师和用户体验人员，还包括技术人员、传统行业的从业者甚至庙宇中的僧人都聚集在这里。因为移动互联网的美妙之处在于它把我们的梦想变得触手可及了。《可可私房菜》的作者是一个热爱美食的女生，不懂技术的她通过朋友帮助将自己的美食博客变成一款应用，并大受欢迎。这样的例子在移动互联网时代不胜枚举。这个时代下消费者的时间、兴趣、记忆、行为等都呈现出碎片化的状态，轻量级的应用易于实施，无论是内容的分享、产品的推广还是弘法布道，只要你有一个好的想法，一支两三个人的团队就可能将梦想实现。

2011年2月《程序员》卷首语“人人都是创业者”中的一段话，可以用来恰如其分地阐述移动互联网时代的特质：“我们正面临永远在线的设备和服务所带来的史无前例的巨大变革，任何现在正在做、想当然的事情，都需要重新思考，寻找适应变革、面向未来的新的可能。也就是说，我们正进入事事皆创业的全新时代。”

无疑，创业道路充满坎坷，但只要怀揣一颗创业的心，积极行动，任何人都有机会立于浪潮之巅。移动大时代已经来临，你准备好了吗？

大会精彩语录



今天有了云计算，有了开源软件，有了销售推广的渠道，让开发和创业成本达到历史新低。

——创新工场董事长兼首席执行官李开复



群众的力量是无限的，这个行业本身是靠颠覆取得成功的，所以对于我们这样一个老革命或者负担很重的公司其实压力很大。

——腾讯高级执行副总裁刘成敏



每一天所有的移动用户在我们开发的应用上消耗的时间是10万年，我想每一个开发者有责任让这10万年的每1分钟过得更有意义，也更快乐一些，这是我们的使命，也是我们的抉择！

——CSDN创始人兼董事长蒋涛



核心难题在于怎么样团结全球的开发者、消费者，造成一种供需两旺的局面，如果没有这个局面所有的其他东西都是那两个字——扯蛋。

——华为终端公司CMO徐昕泉



移动互联产业融合了三个产业的优势：第一个是PC产业强大的运算功能；第二个是互联网强大的沟通功能和联通功能；第三个是无线通信里强大的移动功能。

——微软全球资深副总裁张亚勤



现在每天都有成千上万Android系统的手机和电脑相互连接起来，它对于开发者和供应商来说都是非常好的机遇。

——Google Android全球合作总监John Lagerling



更重要的是一次性开发应用的SDK能不能简单到普通人都会使用，普通人都会使用那就能让更多人做开发者，更多人开发更好的应用。

——中国移动研究院院长黄晓庆



我想请大家为那些真正创新的人鼓一次掌，为那些真正创新失败的人鼓一次掌。

——金山软件董事长、小米科技创始人雷军



这就是市场机会，强调更多的受众，从科学化入手，改变很多人对游戏的印象。游戏今年已经进入到相应的瓶颈期了，我觉得下一个发展阶段一定是科学化和人性化的游戏能占到一定的市场份额。

——顽石互动董事长兼CEO吴刚



你可以丢手机，但你永远不可以丢数据，你永远不会跟他人分享一个手机，但你愿意与之分享手机上的数据。

——阿里巴巴集团首席架构师、阿里云总裁王坚



91%的中国移动网民在用2G网络接入，从细分的高端智能手机上看，Android和iOS也有一半的用户仍然用2G网络接入，四分之一用WiFi，四分之一用3G，这就是我们今天互联网的现状。

——百度技术副总裁王劲



我们有十多亿的生意抛开不做，反倒去市场找一个很好的战友，穿起战炮重新向这个市场进军，这才是活生生的创新。

——诺基亚大中国区总监黄耀丰



移动互联网是一个非常快速增长的行业，我觉得最主要的是移动网络，特别是宽带网络的快速增长，我们已经达到了3G时代，但很多国家已经快速的地建立了4G网络。

——摩托罗拉副总裁沈斌



2011年度第二季度业绩我们刚刚公布，当中社区游戏数据占了80%以上的份额，上个季度这个数字是70%，也就是说我们越来越成为一家依赖于社区游戏成长的公司。

——DeNA中国区CEO王勇



社区化的游戏肯定是未来发展的趋势，因为社区化是解决营销问题的好方法，它能够借用一些交叉的营销把新用户带进来，同时增加用户黏度，进而增加消费频率。

——第九城市副总裁沈国定



移动应用在企业级应用方面可以注意三个区块：移动平台本身、移动应用、移动服务。

——SAP中国研究院总裁芮祥麟



云真正的意义在于它的交互，在于它的精准，在于它的同步分享、分析，还有它给我们带来更低的成本。

——网龙公司副总裁胡泽民



手机娱乐、手机移动社交游戏，应该做很多在互联网上完全做不到的、完全不敢想的，而不简单是一个把现有的农场游戏或成熟游戏移植到手机上。

——Zynga中国区总经理田行智



Web App下一步的发展，第一是会充分发挥HTML5优势；第二是会充分发挥移动交互优势；第三是需要一个能够充分和本地应用匹敌的App，性能和应用都可以全面和本地应用抗衡。

——百纳信息CTO刘铁锋



Windows Phone开发主要问题在于必须要和信用卡绑定，不支持任何其他的支付方式，对开发者影响最大的是什么？不能够支持in-APP Purchase，美国用户更多喜欢先尝后买的方式，而中国消费者不同，这是目前影响获利的最大问题。

—— OpenXlive中国区CTO马宁



小公司执行力非常强、沟通成本较低、自由度较高，但同时也缺乏资源、用户认知度低，出路在于树立品牌、做不一样的事情、保持快速前进。

——eico design创意总监张卷益



百度和Google是两个非常便利的搜索引擎，如果你现在需要的搜索是输入自己的关键字，但必须把自己的特性作为关键字，这样才能提高搜索的出现率。

——App Annie公司CEO Bertrand Schmitt



2011年11月3日，移动开发者大会TOP30+晚宴，施炜亮、陈昊芝、刘铁锋、汤仲宁、William Wei、叶舟、刘泱、PeakJi、江涛、葛楠、崔浩、王智罡、梁公军、高重建、杨祥吉、雷永开、吕淼、李幸福、刘琨……每一个响亮名字的背后都有一段传奇的移动应用的故事。今天这些Legend级的中国移动开发者领军人都会集到了一起。



移动开发者大会还举办了九城开发者欢迎酒会、创新工场沙龙、DeNA开发者沙龙、开发者狂欢夜等一系列丰富多彩的晚间活动。

创新应用展也是本次大会一大亮点，CMDN对30个创新应用进行了评选，吸引了大量开发者参与。





施炜亮
中国最成功的iOS团队之一Triniti Interactive CEO，成功推出多款App Store上榜产品。

App Store式失败之八大经验

演讲者 / 施炜亮　整理 / 陈秋歌

在本次移动开发者大会上，施炜亮谈的不是成功，而是聚焦于他们研发的失败产品，并一一分享从中获得的经验。这些经验包含更为宝贵的真知灼见。

失败是每个公司都必须面对的事情，甚至可以说失败是所有公司最擅长的事情。如何应对失败与公司的战略及战术有很大关联。Triniti Interactive致力于成为国际化的游戏公司，在战略上会积极做一些新的尝试，在战术上会控制好单个项目不必要的损失。我们不惧怕失败，并且会再次尝试失败项目，从中找出失败的原因，进行总结。

失败经验一：不要增加游戏空间的复杂度

《Ancient War》是一款策略类游戏，结合了即时战略以及养成要素。起初这款游戏还比较成功，在之后的更新中，我们添加了飞行系统，以使游戏更好玩、更耐玩。结果我们发现，增添了飞行功能后，地面部队和空中部队之间产生了交互，使整个游戏变得更加复杂，可以说是盲目复杂。于是，用户就表现得手忙脚乱，有很强的挫败感。

针对这款游戏，我总结出：深化一款游戏，要从同一方向入手，而不是盲目添加平行线。随意增加平行线往往会使问题复杂化，也会使用户更加困惑。正确的做法应该是深化、丰富游戏内部新元素，尽量不要在两个平行空间上进行变化。

失败经验二：细节决定游戏品质

《Spartans vs Vikings》是一款即时战略动作游戏。该游戏设计了比较复杂的游戏情节，囊括了战术、战略、战役三个层面，属于综合类游戏。它由新团队负责开发，但其中忽略了很多细节问题，粗糙的设计，最终导致了本游戏的失败。细节决定游戏品质，这是我从该款游戏的失败中得出的结论。为此，我建议一款新游戏最好由新老成员组成的团队来开发。团队的综合素质往往是能否做出优秀产品的最大挑战。

失败经验三：重视市场和用户的差异

《KungFu Warrior》是一款横版闯关游戏，玩家在游戏中扮演一个类似李小龙的功夫高手。该游戏的细节效果恰恰与《Spartans vs Vikings》相反，在制作品质、动画表现、操控等方面做得十分出色，而美中不足的是假设太多。我们在游戏中做了很多假设，比如假设玩家都认识李小龙；假设玩家都喜欢格斗的感觉；假设玩家都喜欢多段联机。而在游戏测试时，我竟然连第二关都没通过。

最后通过对该游戏的总结，我们得出问题出在很多“假设”上，这些假设徒增了游戏的难度。从这款游戏开始，我们也越来越重视市场和用户的差异，不随意做任何假设，你喜欢的东西并非所有人都喜欢。

失败经验四：注重游戏的社交性和游戏性

在开发电子宠物游戏《iPuppy World》时，我们又犯了一个错误：过于注重宠物的

表现，而忽略了它的社交性和游戏性，最终导致该游戏从App Store排名18的位置彻底消失。

产品的核心应该定位于它的社交性和游戏性，而不是它的表现力，这是该失败经历带给我们的经验教训。

失败经验五：不进则退，成熟之后更需要变革

系列产品《Ada’s Series》包含7款小游戏，分别为有艾达的医院、幼儿园、匹萨店、农场、超市、健身中心、时尚秀。当时在我看来该系列产品已相当成功，但在7个系列产品发布之后，用户群却没有太大变化，甚至有缩小的趋势。所以，在游戏行业迅速发展阶段，如果仅停留在原有方向上，维护现有用户群，那么死亡只是一个时间问题。我们首先采取的措施是暂停该系列游戏，并把团队成员分配到其他项目组，让他们在不同领域进一步学习，以期在当前领域有进一步的突破，设计出和之前系列完全不一样的产品。因为我们知道不进则退，成熟之后需要更大变革。

失败经验六：注重多角度创新

2009年，我们发布了一款3D射击游戏《AC-130》，并一举获得成功，甚至压倒了国外同类产品，得到用户的极大好评。当时考虑到该游戏类型单一，没必要再精细下去，于是便放弃了继续研发《AC-130-2》，转而进一步开拓其他未知领域。然而就在今年，一款改良版的《AC-130》横空出世，并挺进了App Store前十名。它在《AC-130》的基础上进行了多方位的变革，它的创新远超《AC-130》。这件事对我的影响很大。只考虑《AC-130》的延伸，而忽略了对其进行变革，是我当初的失误。

对于一款游戏，应该从多角度考虑创新，可以改变题材、改变模式，让它更适应现在的时代和现在的用户。

失败经验七：在决定性领域全力以赴

最初感知到游戏将向社交这个大方向发展时，我们就开始着手研发大型网络游戏《Call to Arms》，并且在2009年6月发布了第一个版本。我们差不多是第一批在App Store开发网络游戏的公司，但我们并未对该类型的产品显示出足够的信心，也没投入较大的精力。一心钻研于其他领域的同时，错失了整个市场变革的大好时机。

这件事给我的教训就是睁大眼睛、预测变化，在起决定性的领域全力以赴。

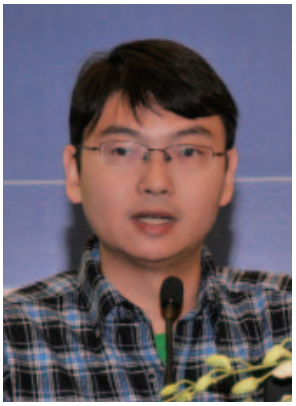
失败经验八：提前筹划产品的变革与升级

《Gamebox》是我们公司中期的一个核心产品，融合了58款游戏，整体取得了较大成功。在最初我们就预测到可能会有很多的模仿者，所以在研发的中后期便大力提高该产品的质量，可以说它在质量上压倒了众多竞争者。2010年3月，Gamebox技压群雄，在App Store上排名第一。

《Gamebox》虽然很成功，但仅仅是战术上的一种成功，它同样也暴露出一些问题：随着新游戏的不断加入，用户对每个游戏的关注度开始下降，同时它的客户端也越来越大，以致最后达到了300MB以上。而我们当时的工作重心只放在增添新游戏上，而没有进一步地加强现有游戏的趣味度。以致在设计跨度一年之后，对《Gamebox》的升级造成了很大负担。

我们本应该在《Gamebox》开始设计的3个月后就考虑产品的变革与升级，更多的注意力应该放在现有产品上。所以在《Gamebox》之后，我们进行了长达一年的痛苦转变，不再频繁推出新产品，而是逐步提高产品品质，提高核心竞争力。

最后向大家提一点建议：不要过度依赖某一种战术。就拿《Gamebox》来说，它的战术实际上只能使用一次，不能再用第二次，不及时的战术手段可能会使你丧失长期的竞争力。现在游戏行业还比较浮躁，大家一定要耐得住寂寞。来日方长，一款成功的游戏需要拥有持久的生命力。



刘铁锋
百纳信息技术有限公司CTO，W3C标准化组织成员。带领团队开发的海豚浏览器在CNET评选的2011年全球最受用户欢迎Android应用中排名第二。

移动设备上的 Web App历史以及发展

文 / 刘铁锋

尽管HTML5的标准尚未完全落定，但主流浏览器已经开始把对HTML5的支持作为了主要的卖点之一。在移动设备上各种框架、解决方案开始逐渐兴起，为移动设备上应用开发带来更多的想象空间。作者在本次移动开发者大会上，深入分析了Web App的发展前景。

移动设备上的Web浏览标准发展历程

正如罗马不是一天建造出来的，移动设备上的Web标准以及应用的发展也同样如此。追溯到智能设备的发展，需要把时间线拉回到10年以前。

如大家所知，作为现代浏览器的标准，HTML最早在1993年通过草案的形式予以发布，到1995年成立W3C之后，才逐渐推动成型。到2000年HTML4.01发布之后，HTML的标准一直都没有太多的改动。而被手机浏览器上所使用的WAP标准，并没有比HTML的标准晚太多问世。早在1997年6月，WAP（无线应用协议）就由摩托罗拉等厂商制订成了移动电话、PDA甚至于未来家电之间所进行通信的标准。移动设备上的网页浏览应用则随之兴起。

智能设备同样也是伴随着标准的推出而蓬勃发展。早在1996年，Palm（没错，就是被HP收购的开发webOS的厂商）就已经开始推出手持智能设备，不过流行的名称是PDA（Personal Digital Assistant）。尽管早期的Palm上没有独立的网络连接设备（比如WiFi/GPRS模块），但Palm的发烧友通过一些设备（比如Palm TT、Palm TE等）上的蓝牙功能绕道，连通手机的GPRS进行网络访问。这是最早期的Palm上的访问曾经流行一时的Palm Infocenter的截图。



通过移动设备访问网络，显然不仅仅依靠使用Palm这样发烧友的推动才能兴起。恰恰相反，正是因为在国内有很多人没有条件使用PC，而只能使用手机来打发时间。在这样的背景下，2006年左右，WAP开始成为国内的一个热门话题。

在这个转折点上，不同的技术方案和思路，造就了不同的结果。有人认为手机

因为处理能力有限，WAP将大力发展，甚至会超过Web网站的数量；有人认为Web上的网站必将成为主流，应该通过技术的手段来把Web的页面转换到手机能够支持的格式。正如大家所熟知的，UC浏览器正采用把PC上所浏览的Web页面通过压缩的方式转换到移动设备，从而赢得了众多用户，而更多仅仅支持WAP的浏览器则早已不知所踪。与此同时，毫无疑问，终端设备上Nokia的智能手机一直雄踞市场老大地位，几乎无法被撼动。

但技术的发展，总会改变市场的格局。正如Google利用搜索，迅速改变了互联网的格局，打破了PC上微软强势的格局一样。手机设备的发展，也改变了WAP的发展以及Nokia的命运。

智能手机上的Web发展

如果没有iPhone的推出，也许智能手机还会停留在分辨率在320×240、必须拥有键盘等等状态；技术的创新可能只是停留在硬件的创新，比如推出更薄的手机、更加具有金属质感的手机等。而iPhone所代表的新智能设备，快速地改变了这一点。用重新发明了智能手机来评价iPhone并不为过。不仅仅是没有键盘、更大的屏幕、更高的分辨率，更加重要的是在iPhone上可以不做任何压缩，直接访问适用于PC上的Web页面。而这一点，为Web在手机上的发展打开了一个新的局面，手机已经可以直接处理PC的Web页面。那么对于手机上是Web会更加占有优势，还是WAP页面更加具有优势这个问题，对比一下当前Nokia和Apple的发展状况，结果已经不言而喻了。也就是说，对于WAP这个过渡技术来说，在新的手机发展阶段，将会慢慢沉寂下去。与此同时，手机上的Web技术发展将会逐渐升温。

谈到手机上的Web技术发展，不可避免地谈到一个问题，就是对Flash的支持。如同大家所知道的，Adobe已经公开宣布放弃手机上的Flash，转向支持HTML5。这的确有点壮士断腕的意味，当然，这并不仅仅是因为乔布斯的个人恩怨而使得Flash不能出现在iOS上，从而导致了Flash的没落。

Flash究竟为浏览器带来了什么？Flash本质上弥补了浏览器在交互、视频播放、本地处理能力的方面的不足。可能大家还能够记得起在2004年曾经风靡一时的小小系列，几个简单的小人就能够演绎一把功夫大片。这已经足以说明Flash对浏览器的增强和补充。如果Flash要更进一步， 那就会更多地接管和服务器端打交道的数据处理工作，侵蚀ASP.NET/PHP等成为前端开发的主流技术。在2007年推出的Flex正是由此而生，期望能够全面接管Web上的开发。如果这件事情成功，大家可以想象Adobe将能够有何种成就，甚至不排除可以全面侵蚀甚至垄断浏览器市场，自己定义出新的标准。如大家所知道的一样，微软很快出手推出了Silverlight，希望能够在这个领域有所建树。也如大家所知道的，由于浏览器版本的分裂，Firefox的异军突起，以及Chrome这样的新秀杀入，谁也没有办法一统江湖，再加上开发者社区推广等种种原因，Flex以及Sliverlight也并没有得偿所愿。

但浏览器上需要增强交互、视频处理、本地处理能力等刚需依然存在 因此，HTML5的兴起，以及微软、Google、Apple的鼎力支持以及大力推动，自然不难想象其背后的原因。在浏览器的重要程度日益增加的今天，各家厂商都希望能够自己成为标准中的一席，然后齐心协力把Adobe的重要位置给挤开。

在HTML5提出的新特性中，不管是新的video表现，还是增强的拖拽功能，还是本地存储功能，本质上都具有了取代Flash的能力，并且甚至有可能做得更好（比如通过WebGL来实现3D绘制）。

Web App与Native App之争

在HTML5升温之后，不可避免地出现了Web App和Native App的争论，谁将赢得天下，谁将成为主流？如果对比一下历史，将会发现，历史早就作出了判断。

回顾一下业界技术发展的趋势，在目前是互联网的公司的规模更为壮大，还是桌面的软

件开发公司的规模更加壮大了？毫无疑问，互联网公司成为主流，在规模上，数万人的公司规模也不足为奇。而传统的软件开发公司则很难成长为这样的规模。原因何在呢？

我们从技术和商业两个方面来进行对比。

从技术方面来说，汇编/C/C++/Java/Delphi/PowerBuilder/.NET一直占据着桌面软件开发的主力战场，尽管开发的难度逐渐降低，但是程序员不可避免地会碰到内存问题、性能调试问题，通过P/Invoke或者JNI等方式来调用系统函数，甚至于各种各样的诡异语法细节问题。对于任何一个开发者来说，如果不能够在语法、库函数调用、操作系统函数调用、各种网络请求调用、多线程问题、内存泄漏问题等方面有深入心得，没有3-5年的工夫，很难完成大型系统的开发。而对于Web的技术开发来说，使用HTML+CSS足以有丰富的表现力，JavaScript的语法和调用相对直接，哪怕服务器的后台，PHP/Python有大量的库来帮助开发能够快速地搭建系统。暂且不论在部署/升级的便利，从技术上手的难度来说，Web开发的技术要来得更快一些，开发人员也相对更多。因此相对来说Web的技术发展也更为广泛，导致越来越多传统软件搬到了Web上。

从商业的角度来说，成本会是一个非常重要的考虑因素。不管是B/S更多地取代C/S的模式和架构，能够减少部署的成本，还是因为有更多的Web开发人员，这样能够更加容易找到员工，Web技术相对更加成熟的框架和应用能够部分加快开发速度。总之，出于减少成本的考虑，不可避免地Web技术获得了更多的支持，不仅仅是互联网企业，传统的企业也更多地选择了Web技术。

那么回到目前移动互联网上所面临的问题，不仅仅是开发速度的问题，还面对了多个平台、多种机型、升级部署等一系列问题。哪怕仅仅是为了测试，也都会花费非常多的代价。那么，有多少非主营IT企业，能够承担得起雇用和寻找能够横跨多个开发平台的专业人才以解决技术问题的需求？因此，在移动互联网上，企业也会积极寻找低成本的解决方案。

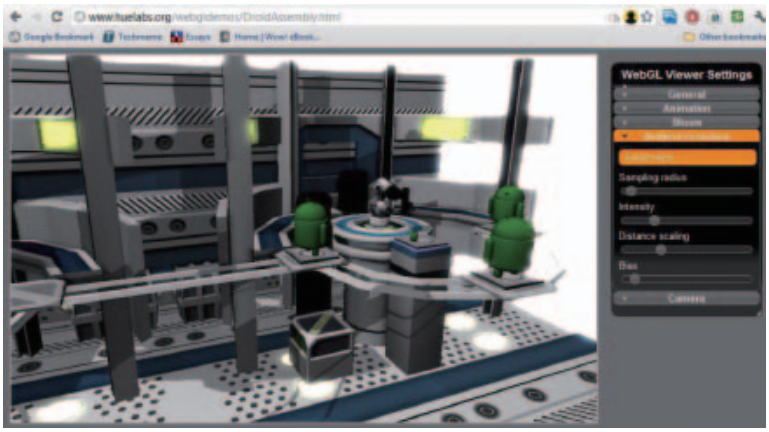
从技术上来看，HTML5也提供了可行的技术基础。这里有一个可能不是严格贴切的对比。（“?”代表不完全贴切）

HTML5让Web App能够提供类似Native App的功能

	Native Code	HTML5
界面的表现方式	√	CSS 3？
界面的渲染方式	√	HTML？
与桌面的交互处理能力	√	Drag & Drop
多线程并行处理能力	√	AJAX？
本地离线存储能力	√	Web Storage/Web SQL Database
访问本地硬件资源能力	√	WebGL

从上表可以看到，HTML5已经让Web App提供类似于Native App的开发能力，从而使得在移动上设备商能够更快地开发出可以和Native App媲美的程序。

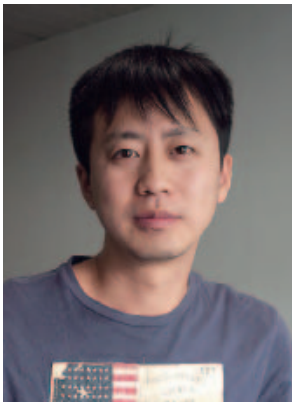
因此，结合技术发展的历史以及当前HTML5的发展趋势，可以看出，移动互联网上不可避免地会复制PC上已经出现的模式。这也是为什么Web App开始日趋重要，会有更多的厂商推崇Web App的原因。



HTML5游戏

结语

时至今日，Adobe已经开始宣布在移动设备上全面支持HTML5。Facebook同样也在提供HTML5的开发平台。不仅仅如此，类似于Finance Time这样的Web App，已经开始能够提供媲美原生程序的体验。可以想象，在未来的几年中，随着Web App的发展，HTML5技术的推广会更加迅猛。



廖巍
Seeed Studio市场总监，致力于推动创客文化以及开源硬件的发展。分解控，尤其喜欢拆电子产品。理想是将“中国制造”变为“中国创造”。

终于轮到开源硬件了

文 / 廖巍

Seeed Studio是目前国内发展得最好的开源硬件厂商之一，廖巍作为嘉宾出席了“2011中国·移动开发者大会”，并撰文全面阐述开源硬件的发展。

“开源硬件是指所有设计资料都公开化的硬件。任何人都可以学习，修改，传播，制造和销售这些设计资料以及基于这些资料的硬件。”

——《Open Source Hardware (OSHW) Statement of Principles 1.0》

疯狂的开源

是的，“开源”听起来很疯狂。相比大公司绞尽脑汁巩固技术壁垒、专利战争打得热火朝天，开源这种自愿被扒光洗净、任由宰割的做法，简直傻得一塌糊涂。但真的是这样吗？

回到1991年，当Linus Torvalds发布Linux时，大家也认为开源软件是件很疯狂的事。没人相信一群业余爱好者能做出操作系统这么复杂的东西。即使做出来了，大公司也不会冒险使用这种“出了问题都不知道该找谁解决”的软件。

历史否定了这些论点。

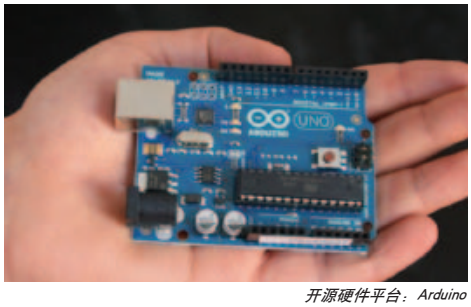
分享是前进的基石

开源硬件和闭源硬件的特征对比：

	闭源硬件	开源硬件
所有权	企业、受到版权、专利、商业秘密等保护	设计者，保留使用版权的权利
再开发	限制	鼓励
开发工具成本	高	通常免费
开发动机	商业需求	设计者的创意
用户活动	使用，但不知道原理	任意使用、学习、修改、传播

硬件设计、特别是数字电路及芯片设计，往往具有很高的可重复性。开放设计文件对个人而言，意味着能方便地进行学习和借鉴。而对于企业用户，则意味着能方便地实现定制和重构，从而提高设计效率、降低研发成本。

更重要的是，和开源软件一样，开源硬件也拥有强大的社区支持。社区成员们乐于分享自己想法、技术和知识。良好的交流氛围，有助于打破设计思想的闭塞状态，促进技术的进步。这点在Arduino身上有着很好的体现。



开源硬件平台：Arduino

“Arduino是一个开放的硬件平台，包括一个简单易用的I/O电路板，以及一个基于Eclipse的软件开发环境。Arduino可以被用来开发能够独立运行，并具备一定互动性的电子作品，也可以被用来开发与PC相连接的外围装置，这些装甚至还能够与运行在PC上的软件（如 Flash、Max/Msp、Director、Processing等）进行沟通。”

——《Arduino中国》

这个最经典的开源硬件项目，诞生于意大利的一间设计学校。Massimo Bazi先生的学生常跟他抱怨找不到便宜又好用的微控制器来驱动他们的机器人。Massimo和朋友David Cuartielles讨论了这个问题之后，两人决定自己设计电路板，并邀请学生David

Mellis设计编程语言。一周后便设计完成。这个电路板被命名为Arduino。名字来源于附近一个学生们经常去的酒吧。

聪明的Massimo意识到开源可能会引发病毒式传播效应，给Arduino带来更多的关注。所以，他将所有设计资料都发布到了网上。事情发展真的就像Massimo设想的一样，短短几个月时间，Arduino不仅曝光率飞速增长，而且热情的极客们还就电路和编程语言的改进，提出了不少建议。Arduino Team利用这些免费的工作，使得自己的设计得到了极大的提高。

开源硬件不是免费蛋糕

免费无疑是开源软件最吸引人的地方。但硬件毕竟不同于软件，硬件的制造和传播都需要花费真金白银。不仅如此，开源硬件厂商还需要制定合理的价格以获得足够的利润用于支撑自身的运营。所以，硬件开源并不意味着大家可以免费获得硬件。

那些开源硬件引发的创新

看看大家都玩出了什么。



1 Makerbot——将想象化为现实的3D打印机
3D打印是一种添加制造技术，通过一层一层的黏合物质来制造三维物体。通过它，使用者可以实现快速、简单、低成本的模型制作。

2 Gadgeteer——为.NET软件编程人员准备的硬件开发平台
微软的.NET Gadgeteer是一套开源工具包。开发者可以使用.NET Micro Framework和Visual Studio／Visual C# Express轻松实现硬件操控，从而快速、简单地完成电子项目设计。

3 DSO Quad——装在口袋里的数字示波器
仪器小型化的杰出之作，能满足基本的电子作业需求。

4 DIY Drones——让普通人也能拥有自己的无人机
DIY Drones由著名的长尾理论作者Chris Anderson创办，以销售DIY无人机模型的配件为主。

5 Lilypad——让衣服亮起来的可穿戴式计算机
Lilypad是Arduino的一个变种。它小巧的外形设计以及丰富的外围模块配备，使其成为目前最为流行的可穿戴式计算机。



2011中国·移动开发者大会的开源硬件展

新的生产方式

绝大部分开源硬件产品并非系出名门。这是因为开源硬件产品多是源于设计者的自有创意，非基于传统的公司市场调研报告。同时开源硬件原理公开化、开发简单化、以及社区协作化等特点，为设计者实现创意铺平了道路。

越来越多人为了满足自身的需求，开始自己动手做东西。以草根创造为核心的创客文化也随之诞生。想想给自己做个端洗脚水的机器人，给女儿做棵会唱歌的圣诞树，给老婆做串会发光的项链，是不是很Cool？如果，还有很多人想要拥有和你一样的东西。

恭喜你，从此你可以走上一条全新的创业之路。

那么怎样开始?

创意：无论硬件和软件，都只是表达思想的工具。创意才是一切的灵魂。每个创意都源于一个实际需要解决的问题。所以，你需要拥有一双发现生活的慧眼。

设计：兴趣永远是最好的老师。带着兴趣寻找志同道合的人。无论是线上社区还是线下创客空间，只要愿意分享你的设计，相信你能收到很多改进建议。反复打磨你的设计，让它达到一个让人满意的程度。

资金：到“点名时间”这样的资金众包平台发布自己的项目。在募资的同时，也可以对市场进行前期测试。

生产和销售：硬件的生产需要经历一系列复杂的过程，从PCB制造、元器件采购、PCB焊接，到测试和包装。对于个人或小团队而言，自己来完成这一过程是很困难的。同样，产品的销售、收款、物流和售后服务也不是件轻松的事。对此，Seeed Studio提供了微生产和代销服务，将自身的生产、销售资源与开源社区进行最大限度的共享，使设计者能将精力集中于设计以及社区支持。

竞争不可避免

公开设计资料从某种程度上意味着主动引入竞争。当你的产品卖得好时，仿冒必然随之而至。这时，品牌和社区将是你有力的防御工具。质量和服务使你的产品能区别于众多廉价低劣的仿冒品。

同时，绝大部分社区用户也懂得保护生态系统的重要性，不会倾向于购买仿冒品。不仅如此，他们源源不断输出的改进建议，让你的产品能始终保持领先。

圆桌论坛：移动互联网的创业与投资

记者 / 张宁

移动互联网为创业者提供了无限的想象空间。然而面对移动投资这片创业红海，投资者如何选择投资方向和创业团队？如何评估在移动领域的投资回报？对移动互联网未来发展方向如何预测？这些都是需要深思的问题，本次移动开发者大会，动点科技TechNode创始人兼主编卢刚就此话题主持了一场精彩纷呈的圆桌论坛。

要具备全球视野

Kelly Poon：ATOMICO于2006年成立，在几年时间内投资了50多家公司。中国是一个不错的市场，我来北京6个月，就投资了两家公司。虽然在中国竞争很激烈，但作为外界投资者，我希望看到更多真正帮到用户的创业项目，真正帮助有创业心的人。我们公司的大方向是：希望在中国挖掘一些团队，将其推广到全世界。我最关注的是中国企业能否在全球有更大的发展。

丁辰灵：我开过互联网公司、游戏公司，从创业一路走过来，现在做一些早期的天使投资。虽然创业需要勤奋，但如果方向是错的，再勤奋也没有用。在正确的方向上进行积累和耕耘都会有所回报。

移动互联网的国界性和传统互联网不同。苹果提供了一个全球的渠道。在这种情况下，中国的创业者可以为全球的用户提供服务。这就要求创业者具备全球化的思维和视野。虽然今天在中国还很难挣到钱，但可以先在外国挣钱，等中国市场成熟时再在中国市场做更多的耕耘。

William Wei：我的大部分经验来自美国市场，两家被收购的公司也在美国。但我在美国的经验没有全部带到这里，我自己也在不断学习。进入中国时是从Online游戏切入，现在我们团队有15个人，在过去6个月内成型，刚刚推出一款原创游戏——《上海1930》。先发行了Android版，可能在下个月推出iOS版。为什么选择从Online游戏切入？因为它的难度最高，游戏团队需要有后端、前端、策划、UI、美术、BS以及数据分析。虽然整个团队的建立很难，但这是对团队最好的历练。

叶舟：永远保持创业者的心态，才能不断进取。我们优选了美国、澳洲和台湾的技术，通过精准体感、多屏无线互联、人脸识别等芯片和硬件解决方案，为移动互联网的开发者提供跨平台应用。我们跟运营商提出把手机和电视无线互联，直接把手机影像投到电视上，手机就是游戏机，手机屏幕可以直接控制电视。CyWee的整个系统架构可以实现在搜索上实时压缩，开发者就不必再管操作系统，让游戏可以跨任何平台。这样就给应用开发商提供了很大的平台。触摸屏是台湾开发的，最后苹果把它发扬光大了，硬件加上应用也可以造成一场产业革命。

移动互联网要看长期回报

丁辰灵：投资讲究的是抓住机会。比如地铁站已建好，在附近买房子算是有增值空间吗？如果预先知道这个地铁站一年以后会建好，现在我就去附近买房，这才能拥有更多的增值空间。投资并不是今天投资一定要挣到钱，而是看未来是否有挣钱的趋势。

论坛嘉宾


ATOMICO风险投资公司亚太区域总监
Kelly Poon


移动世纪会秘书长
丁辰灵


MoblieAppExpress创始人
William Wei


速位互动股份有限公司总裁兼CEO
叶舟

今天可以不挣钱，但要知道自己在做什么。至少我知道今天第一个应用没有挣钱，但一年下来我培养了团队，这个团队有很好的执行力和成长性，我投资的应用有很多人知道，有很多人相信我们团队的能力，这个团队本身就是增值的。价值可以用很多方面来衡量，能不能立刻挣钱是一种价值，能不能培养团队也是一种价值，只要有价值就不会担心最后没有回报。

Kelly Poon：我跟大家分享一个真实的案例。ATOMICO成立初期，很多有名的创业者来找我们，但我看不到任何盈利点，所以当时考虑了很久。但最后还是决定投资，而且我们还是大股东。因为我觉得作为一个投资者，短期我会看，远期我也会看。

William Wei：首先要看产业的成熟度，现在我看中的是大的方向是否正确，因为没有人能对未来看得很准。其次要看团队，需要什么样的团队要经过市场的反应决定，团队要能快速满足市场的需求。

叶舟：有人认为移动互联网的趋势是由硬件驱动而不是由服务驱动的，我认为有些道理。拥有解压缩能力的芯片，实现了手机与电视机互联，为游戏提供新的应用思路。领先的体感方案，在手机里添加陀螺仪，使得在商场里能做到精确到一米的地图搜索。同时服务也很重要，UI的不断修正、用户使用的便捷程度都需要长期完善。

产业过热？给移动互联网创业者提一个醒

Kelly Poon：我个人认为这个产业太热闹了一点。国外的投资者在中国创造了很多机会，但不要让市场热潮改变了你，永远要记得你想做的是什么。创业者要牢记这一点，我们投资者也需要看到这一点。

丁辰灵：大浪淘沙，这个浪潮在互联网时代经历过，在Web2.0时代也经历过。其实最重要的是整个过程中能否学到东西，同时能否享受自己做的事情。

William Wei：有人问我在硅谷创业和在北京创业有什么不同。我的感觉是在北京创业，在硅谷创新，在美国不是开发者要创业，而是想要解决一些问题，不管是自己创业还是给别人打工，市场上没有的就把它做出来。但这种情况在中国比较少，大部分是开发者想创业，在找项目。

结语

目前有大量的开发者会加入到这移动大航海时代中，而谁能找到真正的宝藏，还要倚仗稳健的创业心态和全球化的视野。

“建立企业中的用户体验力量”系列之四

用户体验的跨职能协作

文 / 吴卓浩

既然用户体验工作无法由用户体验人员单独完成、而是由整个大团队中不同职能的人员共同完成，就一定会遇到协作的问题。如果关起门来，分别让产品经理、设计师、工程师召开本职能诉苦批判大会，你会听到以下内容：产品经理说设计师和工程师没有大局观、总在一起掐架；设计师说工程师常借口技术上做不到、不愿做改进；工程师说产品经理和设计师照着自己的想法反复改产品设计、找麻烦……类似的情况在各种各样的团队中都出现，是因为有些员工性格或者共同交流的软技能不够好造成的吗？可能有这方面的原因，但显然另有其他主要原因。

以优秀用户体验为团队的共同目标

各个职能的人员从各自的专业背景出发，完成本职能的工作，大家都为产品作出了自己的贡献。图1这幅漫画表现的是不同职能的人员如何构思同一款产品，以及把这些构思合在一起产生结果。

虽然说“每个人心中都有一个自己的哈姆雷特”，但对于产品来说，团队没有一个共同目标的结果是灾难性的。类似“做最好的产品”这样的话是无法成为实际的共同目标的，因为不清楚要把什么做到“最好”、“最好”的标准是什么。当把你的所有目标都放在一起，把其中最不重要的一个目标减掉，最后剩下的那个就应该是你团队的共同目标。比如在做一个艰难的产品决定时，所有的目标和条件都拿出来PK，到最后一刻，有的企业以是否对用户有益为最终标准，而有的企业以是否获利最多为最终标准，还有的企业以是否让大老板满意为最终标准，这反映的就是这家企业的共同目标和核心价值观。正如前面曾探讨过的，

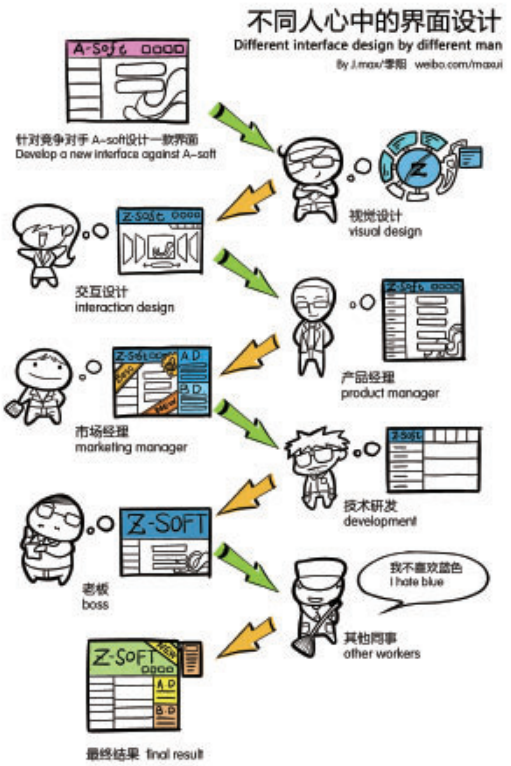


图1 不同人心中的界面设计By J.max，Weibo.com/MaxUI

用户体验的价值比较难于衡量，我们无法期望每个团队都自觉地将产品用户体验优秀作为团队的共同目标，但我们可以看到，那些取得真正巨大和长远成功的产品，往往都是真正重视用户体验、以优秀或独特的用户体验为目标和核心竞争力的。

当然，团队在产品上的共同目标在不同的产品阶段也需要有不同的侧重点。有的产品，在产品起步期，侧重在获取新用户，通过简洁友好的用户体验快速增长用户量；在产品成长期，侧重于发挥高级用户的能量，通过逐渐丰富的用户体验引导用户发表更多高质量的内容、发起更多的用户间互动；在产品成熟期，侧重于维持和转化用户，通过提供差异化的用户体验让不同类型的用户之间不互相干扰、鼓

励初级用户向高级用户转化。你可以找到适合自己产品的用户体验侧重点和衡量标准，建立起自己的具体化的共同目标。

以优秀用户体验为整个团队在产品上的共同目标，是保证产品以最佳状态出现在用户手中的最有效的方法，但这种改变在现实中往往对已有的团队带来很大的冲击。因为大家的工作习惯已经形成，可能习惯于先做出一个很酷的技术、再想对用户有没有用；可能习惯于按照自己的设想做产品、不想用户需不需要；可能习惯于照搬国外类似产品的做法、不论适不适合本土用户；可能习惯于不依靠产品质量、而更多靠运营获取成功；可能习惯于老板或者客户要什么就给什么、就不用担责任……要改变这些习惯是非常难的，既要有高层的支持、合适的团队机制推动（比如工作流程、考评和激励制度），也要求用户体验工作的带头人本身具有然人信服的专业能力，并且团结有着相同信念的人坚忍不拔、坚持不懈地去影响和帮助更多的人。

举例来说，工程师抱怨设计师动不动就要改设计，设计师抱怨工程师不愿根据设计改进产品。要想让工程师真正理解和支持产品经理和设计师在产品设计上的决定和反复改进，最好的方法就是请他参与到产品设计中，经历思考分析、设计探索和快速检验的过程，在这个过程中理解设计决定的背后的原因和意义，并且对设计、产品经理和设计师产生信心。在这期间工程师也经常能从实现的角度对产品提出重要的建议，这样对产品设计的建议当然是越早提出越好。同时，产品经理、设计师和工程师也需要一起讨论，怎样实现产品才能更方便灵活地进行以后的改进调整，怎样实现产品是最有效率的。你会发现，这样坚持不懈的结果就是你会拥有一个协作无间、具有大局观、执行力强的团队。

平衡个体体验与群体共性

用户体验工作似乎谁都可以发表意见，但不要忘了，用户体验工作的目的是从个体体验出发、找到群体共性作为设计的依据。这并不是说只有用户体验人员对用户体验工作有发言权，其他职能的人员的意见不作数，也不是不同职能之间不应互相干涉。恰恰相反，只要以正确的方式思考，就可以有效的参与用户体验工作；应该鼓励不同职能的人都以优秀用户体验为产品目标、都为用户体验思考，让团队具有更强大的用户体验的力量。

对产品的用户体验工作发表意见的时候，你要思考一下，你是目标用户吗？你的意见能代表目标用户么？有一个笑话， 某技术支持工程师接到一个用户的求助电话，说连不上网了。工程师说，检查一下你的“猫”上的红灯亮不亮。用户很惊讶地说，我家没有养猫啊！读到这个笑话，你可能会笑；但是在实际的工作中，有很多很多人意识不到用户和自己的知识背景、思维方式、行为习惯是不同的。有很多的例子都是开发者觉得自己的产品很好用，可用户就是不会用；有的产品还能通过及时改进、挽留住用户，但很多产品就此消失了。

任何一条个体体验都是一个线索，追寻体验的群体共性的线索。思考表象背后的原因，请目标用户来做测试，都会帮助你更快的找到答案。

创造性与执行力并重

这是一个让设计师很纠结的地方：探索和创造意味着不确定性，没有最好，只有更好；然而产品有发布日期、有研发计划，产品用户体验工作的执行必须是确定的。这一个不确定、一个确定，就产生了矛盾。

尝试很多的解决方案，却迟迟不能选择其中较优的方案，或者执著于细节，就会影响到研发全局。但这种情况往往只是设计师个人的问题，比较好改进。而另一种情况则对全局的影响更严重，就是因为时间紧、强调执行力，放弃探索问题和解决方案，直接提出一个解决方案，马上开工，按时完成。这会严重影响解决方案的质量，甚至导致返工，欲速则不达。但时间限制摆在那里，怎么办？一方面，需要合理规划时间。把产品设计的时间稍微放长一点，表面上看是开发实现的时间变少了，但事

实上由于产品设计做得更好，开发实现的效率更高、完全可以在更少的时间里做出更好的效果。另一方面，需要改变工作方式。比如通常开会时，会上只讨论、会下做设计，过几天再开会，一看设计不对，回去重新设计，下次开会再看，这样反馈周期长、浪费的时间也多；如果是开会做设计，在会上边讨论边在白板上画出界面效果图，有什么问题当场发现，有什么意见当场提，会开完，设计也基本确定，然后会下就是执行，效率和效果要好得多，如图2所示。

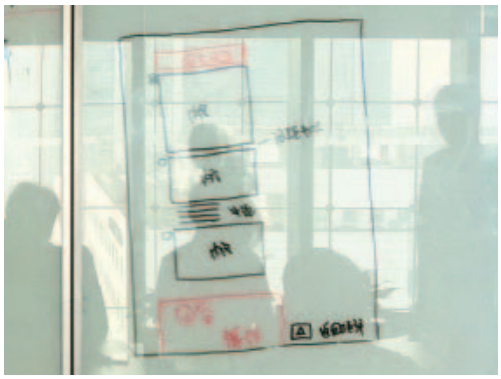


图2　白板文化的背后，是鼓励跨职能协作，鼓励快速设计、快速验证

改善沟通

上面讲了很多工作方式方法，但其实沟通技巧这样的软技能也还是非常重要的。不论是本来就爱拉关系的国人、还是更加独立自我的欧美人，注意沟通方式方法、关注他人的情感和心理需求都是有用的。而且和你一起工作的这些人是你的伙伴和战友，本应成为你的朋友。当然，这方面的书也很多，在这里就不讨论了。

事实上还有另一个层面的沟通，你和合作伙伴说话的语言、使用的工具、做事的方式是否一致，这也是一种沟通。

设计师对工程师说，这两行字之间的距离得增加3像素。工程师那边就有点麻烦了，因为两行字之间的距离主要是由行高、字体、字体大小等几个因素共同决定的，要想在不同的浏览器、操作系统中精确地让两行字之间增加3像素，还真得动点脑筋。但事实上，可能这个行间距并不需要那么精确，其实设计师只告诉工程师，把行高从120%调到130%就好了。

设计师用来做设计的工具Photoshop中，为了方便控制设计效果，可以建立很多“层”分别放置设计元素，还可以建立“文件夹”归类放置“层”。有良好工作习惯的设计师会用文件夹把层分类归纳，并且把文件夹和主要的层都命名。这样，当这位设计师过一段时间再来编辑这个文件或者当几位设计师一起协作时，每人一看到这份文件，就能清楚容易地了解文件的结构，找到要编辑的内容。

设计师通常会给工程师一份设计稿标注图，上面标注了界面元素的颜色和尺寸数值，以便工程师按照这些数值编程实现。有的设计师会在设计图标注大量的界面元素之间的间距数值，这里是1像素，那里是2像素，还有3、4、5、6、7、8、9像素。面对这样的标注图，工程师会苦不堪言，因为需要非常仔细去查看和设置几十处不同的间距。但事实上，一定要区分出1、2、3像素或者7、8、9像素的间距吗？其实，在一套设计中，间距也是一个可以规范的项目，比如一共只定三种间距：小间距3像素，中间距6像素，大间距12像素。这样在设计中就考虑规范，不仅能使设计效果更具清晰的逻辑性，也会大大简化编程实现——甚至都不需要给出标注图，只要告诉工程师这三种间距尺寸。

改善工作上的沟通不只是一项软技能，还有实实在在可以使用的工具和方法。

让不同职能的人员都认识到，产品需要以最好的用户体验状态出现在用户手中，并以此为共同的目标，调整工作的方法和流程，让用户体验思维真正融入到团队的各项工作中，坚持不懈，你就一定会获得优质的产品和优质的团队。



吴卓浩

创新工场用户体验总监，曾创建Google在中国的用户体验团队。本系列文章将分享其在不同环境中建设用户体验团队的经验，讨论如何在企业中建立用户体验力量。

责任编辑：陈博（chenbo@csdn.net）

泛社交化游戏中的平衡问题

文 / 郑金条

游戏的平衡性往往左右着玩家的心理，决定着用户黏性。通过怎样的玩家策略能够保证玩家在游戏中获得良好的体验？本文将对其进行详细的阐述。

均衡属性提高用户黏性

这几天和朋友体验《Street Fighter IV》（iPhone版），感触最深的是技艺精湛的资深玩家也可能折戟于对手的瞬间发力，而类似于我这种在招数配合上略显生疏的玩家，竟然也能依靠简单到极致的手脚动作（非气功或者组合型动作或者绝招）将看起来似乎不可打倒的强势对手消弭于一招一式。

这个层面看起来异常神奇，尽管相比较于在各种招数中游刃有余的资深玩家，新手往往不堪一击，也正是玩家角色的每一次负伤行为并不影响下一次招数的使用和质量，使得玩家和对手之间在每一次的对决上都是站在同一起跑线上，所不同的只是不同角色最后所能承受的重击的次数差异（在《Street Fighter IV》游戏中不能以生命值来判定胜负的可能性，对手任何一个致命性的招数都可能在瞬间扭转整体局面）。

基于《Street Fighter IV》而引申出我们将要讨论的环节，开发者在游戏设计中如何去权衡对决中可能确立的胜负数值，中间是不是多了更多的策略弹性，满足玩家在不同阶段都可能产生的对胜利的诉求，包括高级别玩家和低级别玩家两者之间的对抗如何在优势高下立判的情况下，也能够依照田忌赛马的方式谋得一些胜算的可能【注：出自《史记》所载录的田忌赛马很明确地揭示了玩家策略在游戏中扭转局面的效能，一般习惯中以优势兵力对抗优势兵力在低级别玩家中基本毫无胜算可言的，而孙膑的设想则是以下马对抗上马，即便输得很惨但是却为上马赢中马，中马赢下马创造了全新的机会】。

游戏邦曾在解析玩家的正负反馈中提到过这个问题，玩家战力之间因为布局和策略方面的差异而产生能够改变时局的潜能时，这种弹性空间将使玩家之间的博弈充满不确定性的变数，使两者之间对抗性的进程保持着可能获胜的悬念。这种进程可能会促使两者之间时刻保持警惕和亢奋，随时提防着对手可能带来的致命一击。在《Street Fighter IV》中，使用Dhalsim在和M.Bison格斗时，即使玩家将M.Bison击倒在地甚至生命值已经行将耗尽的状况下，都没什么好值得庆贺的，在M.Bison处于最高难度级别的情况下，一个绝招都能将一个生命值几乎满格的Dhalsim置于濒死的窘境；当然反过来，Dhalsim同样可以在几乎没有生命值的情况下依靠长手的优势置E.Honda于死地。《Street Fighter IV》所赋予游戏的这种均衡属性让原本选择性不多的局面一下子因为变数而变得具有十足的黏性。

依靠适当策略弥补均衡性的缺陷

与此相对应的是玩家具有明显优势的格



均衡性赋予选择性缺乏的游戏十足的黏性



良好的玩家策略有助于改善游戏均衡性不足的状况

局，玩家之间战力或者协同性随着玩家的投入和资源方面的差异（包括虚拟商品的消费）而无限拉开差距，不仅新手玩家，即便是同一阶段的玩家都能感受到这种来自不同层面的压力，两者之间在战斗未开始之前就已区分了胜负。我们曾经以象棋的招数变换来解析这种类型游戏对智慧的魅力，但在均衡性的描述面前，象棋棋手之间对棋艺的领悟和研究已经几乎使博弈的双方缺少可比价值，除非两者能够在伯仲之间还能够在策略上稍微做些布局和影响后续进程。在这种情况下，新手和自认阶段不够的玩家很难能够和高手进行实质性的对抗。一般而言，类似的反馈是相当极端的，因为这种游戏需要智慧涵养，而智慧涵养又确保了这层优势的存在，在既定的事实面前，不会产生任何的对抗乐趣，只有屠戮的惨烈。Brice Morrison将这种情况归属为Slippery Slope（滑坡理论）：落后的格局将恒定落后，并且每一次对抗都能够使弱势一方饱受摧残（象棋具备了易于上手而难于精通的游戏设计趋势），即使通常以让子（比如车、马、炮，围棋在两者实力不均衡的情况下也通常出现让子的局面）的方式来均衡双方的战力，也往往不能弥补双方在棋艺方面的布局领悟力。这和我们之前所描述的情形刚好相反，《Street Fighter IV》玩家即使依赖简单的招式（比如Dhalsim）也可能击败强劲的敌手（比如M.Bison，尽管Dhalsim的长手策略往往不一定能够奏效），与之相似的具有均衡属性的游戏还有《Virtua Fighter》和《Soul Calibur》这种潜在一击定胜负的特征，但象棋的让子行为并不能改变弱势的一方弱的形和神（弱势一方无论选择都已经丧失意义，即便被让子也让进程失去应有的乐趣；强势一方也同样有相似的困境，在实力不均衡的情况下，一样不能有所体味）。

除了以格斗定胜负的因子外，那些依赖成就系统作为用户间参照的游戏也往往在均衡性上进行了调整，Lucas Blair曾以此为参照仔细审视了成就系统对玩家延续可能性的影响（包括自我挑战和玩家之间在数据层面的对抗挑战），可以举一个简单的例子，玩《Angry Birds Rio》的玩家获得一颗星和极低的分数的通关模式显然不符合玩家的心理预期，这时成就系统的效能就是激励玩家向最佳成效发起挑战，而这种自我胜任（好胜）意识驱动了玩家更好地处置和分配优选自己的资源（比如小鸟的个数或者打击的方位），更多地去研究抛物线的路线和最终的结果。因为离开了2+2=4 这种僵固的模式，玩家的游戏进程从原始的盲目投射开始带入了智慧和策略的层面，尽管更加理性化却并不意味着精心准备的一次投射就一定超越玩家随手一击所带来的成就价值高（当然前者的概率属性会相对更高）。

当然也不尽是每一种玩家策略都是上佳的选项（有开发者认为如果玩家能够在游戏中摸索出他们所不曾设想到的方式组合就是开发者最大的惊喜），一旦存在着某种绝对优势的策略，就可能打破我们前文所论述的关于选择所赋予的均衡价值，就如《Tic-Tac-Toe》，玩家可能就会挖掘到整体游戏规则中的漏洞去确保一种高枕无忧的方式，从而为自己固守胜局。尽管这种方式刚开始能让窥探其中的玩家乐此不疲，但长此以往仍然会陷入我们早先所阐述的不管玩家做什么样的选择，对他们来讲已经完全失去意义（他们已经不再能讲述游戏的乐趣，有的只是轻易地胜或者轻易地败）。假设《Street Fighter IV》中的Blanka是永远不可颠覆的，多次的挫折玩家就必然失去继续挑战的勇气和乐趣，而此时一款原本

能精彩对决的游戏就可能因为设置的不均衡而让各方玩家索然无味。

合理运用制衡措施

有人在研究社交游戏（手机游戏）对用户的黏性价值上使用了心理学家B.F. Skinner的Random Reward Schedule来复苏我们刚刚所竭力论述的均衡性，包括弱者和强者之间的差异如何以更合理的方式消弭至少不至于分裂得太明显。《Street Fighter IV》（iPhone版）和《Angry Birds Rio》的模式给定了玩家更好的中间值，而Random Reward Schedule所呈现的是另外一个可能：不定时的随机性诱使玩家在不愿错过的心态驱使下能够重复回访游戏。这和Brenda Brathwaite所设定的让玩家带着期待守着游戏的理论是相似的，随机性的价值赋予新手玩家或者相对弱势的玩家获得补足的可能，比如随机掉落的生命值的补给品，强势的战力装备或者遭遇的竞争对手的级别层次，或者竞争对手出现的时间点或者位置。这种随机化和我们所论述的《Street Fighter IV》在不同玩家之间的权衡上几乎异曲同工。

当然还有开发者使用了更极端的制衡措施，对资源过剩的玩家采取了惩罚行为，使玩家的可有效支配资源控制在一定的幅度内以调控不同玩家之间的差异，比如对商店虚拟商品的购买行为进行限制，使得用户的资源维持在一定的合理范畴内不至于出现一边倒的状况。现在的很多社交游戏或者手机游戏都有这方面的考量，尽量使得用户之间不会出现太大的不均衡裂痕，但事实上这种作用还是相当有限的，对于消费玩家和非消费玩家这种差距依然会越拉越大，这种方式的调节看起来显得不是那么明显。

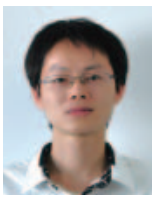
让玩家掌握游戏的主动权

通常还有一种说法，骨灰级的玩家的抗挫折能力更强，可以忍受类似《Ninja Gaiden series》（Tecmo）、《Battletoads》（Tradewest）或者《God of War series》（SCEA）这样难度级别的视频游戏，但一个寻常不玩游戏的玩家可能在《Touchgrind BMX》（Illusion Labs）或者《Canabalt》（Semi Secret Software）这样的iPhone游戏中动摇继续游戏的决心，似乎这已经不是论输赢的问题而是关于玩家能否持续体验的问题。

在休闲游戏中，似乎玩家的承受力稍微脆弱一些，《Adventure World》的多重限制确实阻碍了用户继续体验的勇气。而有所平衡的游戏设计则可能让玩家在游戏中施展得更加尽情，一个玩家掌控着Dhalsim并不意味着就一定能够击败E.Honda，而自以为能够在《Crimson: Steam Pirates》自由划线布局的玩家则可能在搁浅的状况下被围剿。策略层面的布局让游戏充满了维度，这点可以在《Real Racing》或者《Real Racing 2》得到更好的实验，只要玩家有更好的弯道技术，就可能在游戏的圈道运行中占据更好的优势，而任何一个玩家都能够获得这种尝试。Richard Rouse III在《Game Design: Theory & Practice》一书中将这种行为描述为“让玩家获得更多的游戏主动权”，我在体验Gameloft的《Gangstar: Miami Vindication》同样很明显能感触到这一点。

> 玩家战力之间因为布局和策略的差异而产生能够改变时局潜能的时候，这种弹性空间将使玩家之间的博弈充满变数，使两者之间对抗性的进程保持可能获胜的悬念。

让玩家变成一个探索玩家，而不是时时刻刻都处在开发者的布控之下，给予游戏以一定运气成分（甚至偶尔能够改变时局），同样让玩家更多时候处在相对均衡的实力之下对抗，让不同层次的玩家都能够拥有相似的快乐。



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：陈博（chenbo@csdn.net）

也说腾讯应用中心

专访腾讯应用中心商务总监童学红

记者 / 刘明

在互联网上，应用中心已经成为用户心目中下载软件可以信赖的地方。2007年，腾讯也建立了自己的应用中心（当时被称为下载中心），主要提供一些腾讯自有软件的下载。随着腾讯软件应用范围的扩大以及用户量的提高，很多用户呼声很高，不但希望下得到腾讯软件，还希望方便获得其他的软件。为了满足这个巨大的需求，尤其是智能手机日益普及后激增的手机软件需求，腾讯应用中心成立了。这个应用中心做得如何？目前在业界地位怎样？未来的发展计划如何？为此《程序员》采访了腾讯应用中心商务总监童学红，他主要负责与第三方内容提供商的沟通，包括一些内容的引进和审核。以下是对童学红的采访。

《程序员》：请介绍腾讯应用中心是个什么样的产品？

童学红：腾讯应用中心，对于用户来说，是一个能提供很好软件下载的平台。同时，还可以提供其他附属功能，不但可以下载，还能评价软件和看到好友的评价和他下载软件的情况。对于开发者而言，希望能提供一个发布内容的平台，在其上可以发布自己制作的软件。如果软件够优秀，应用中心会给它推荐到好的位置。我们会根据内部系统的算法、用户的行为和反馈进行自动推荐，算出来某个软件应该排在什么位置。将用户最想拿到的应用推到平台的最前面。只要软件做得好，就有好的排名。

《程序员》：应用中心在腾讯整个产品线中处于怎样的位置？

童学红：应用中心是我们内部各个产品的下载平台，也是给外部产品的下载平台。因为中国市场比较特殊，下载软件还有很多第三方渠道，但就腾讯自有渠道来说，应用中心的角色还是非常核心的。

《程序员》：现在应用中心的开放性怎么样？与第三方的合作情况如何？

童学红：有几种合作方式，包括开发者或者内容提供商，不论个人，还是公司，都会有以下四种类型的需求。

第一种类型就是做了一款软件，想放到腾讯应用中心上去，这个基本上没有什么门槛，只要发布的软件合乎法规，都可以在腾讯应用中心的平台上进行宣传、发布。这是最简单的合作模式。

第二种类型是资源互换，是指开发者可以预约我们的一些首发机制和首发推荐位。只要预约之前根据我们的规范，把自己软件的首屏设置成腾讯应用中心，这相当于给我们一个资源；我们经过审核后，认定这款软件合法，并且够优秀，可以满足首发条件，就可以给这款软件一些首发资源。

第三种类型是指有些第三方有钱、有软件，最重要的是还有商业模式，像电子商务、彩票、图画或者影院类的应用，愿意在腾讯广告平台上投放广告，按照下载来付费。目前的结构是按照下载付费。

第四种合作模式是针对希望赚钱的开发者。他们可以接入腾讯的广告平台，以媒体的身份在自己的应用中，嵌入腾讯广告平台的SDK。这样面对的就是腾讯众多的广告主。如果用户对应用内广告点击或者下载，我们将分成给开发者。

如果开发者想要盈利，第四种方式是最好的。目前已经上线一个月，有几百家开发者。他们中每天最高收入的数额，即在这仅仅一个月时间所能达到的成绩，已经基本达到了我们的预期。

《程序员》：和开发者的分成模式如何？



腾讯应用中心11月举行了“开发者/广告主合作巡回沙龙”

童学红：困难就是上线流程遇到过一些挑战。我们是一个大平台，如果包括进一些不良软件，负面效应会扩大N倍，所以腾讯应用中心会更加规范一些。从长远来说，用户会觉得在你的平台下载更放心。同时也给开发者这样一个非认证的资质保证，也就是说，如果在腾讯应用中心已经发布了软件，再向其他渠道宣传，会有一定程度的信用保证。这样对内容提供商、用户和合作伙伴，都是好事一桩。短期来说，有些开发者可能认为我们的进度太慢了，上面的解释也是希望广大开发者能给予理解。同时我们也会尽快、进一步改进和完善整个流程。

童学红：目前是五五分成，但这个模式在年底可能还会做调整，我们正在研究策略，会找到更优惠于开发者的分成方式。

《程序员》：腾讯应用中心如何打造自身的生态链？

童学红：腾讯应用中心成立才半年时间，还有很多东西处在完善过程中。刚才我提到的四点，其实也是一条比较完整的生态链。如果涉及游戏收费，我们会提供一个游戏合作的渠道，这个渠道将提供给内容提供商超过十种以上的支付方式，支持给他的应用完成付费。对于工具类的软件，针对用户的付费模式目前还比较少，这是我们比较少见的需求。

《程序员》：腾讯的应用中心，号称提供全平台的支持，如何保证能有统一的用户体验？

童学红：应用中心会尽量保证统一的用户体验，但因为智能手机的版本型号层出不穷，即使我们在后台完成统一，前端的显示效果也会有些不太一致。所以，有时我们会灵活对不同的平台做一些妥协，比如在Android平台上，实现触屏是很容易的，然而在Java和Symbian平台上基本不可能做到，这里的用户体验从根本上是不一样的。所以我们尽量做到统一的是，用户使用习惯和分享习惯等方面。

《程序员》：腾讯应用中心上线已经有半年，建设过程中，您认为遇到的最大障碍和困难是什么？

《程序员》：如何处理和竞争对手的关系呢？

童学红：其实整个移动互联网，目前都有这么一种情况，也就是一个模式或应用出来以后，有很多人跟风去做。我觉得这个情况是好的，并没有把这看成竞争关系，而是说一起把市场做大。就目前应用中心的领域来说，商业模式还不是很完善。从长期来看，一些参与者肯定要么消失，要么被收购或被整合，最后的场景可能就像现在的搜索市场和电子商务市场一样，剩下几家大型的。现在的移动互联网有点像2000年时的互联网，当时什么应用都有很多人在做，但最后做得很大的可能也就几家。目前大家一起做这个领域是一件共同促进的好事情。

《程序员》：只针对手机应用，腾讯应用中心未来的前景如何？

童学红：我对前景还是非常看好的。手机和PC不太一样，手机界面会很小，但手机未来的使用会越来越多、不管是HTML5这样的技术是否能够大规模普及，手机端的应用都会越来越多，如何满足越来越旺盛的应用需求，提供软件下载的平台就显得越发重要。而且由于手机的计算和存储的局限性，从平台上下载软件的需求会越来越多。

淘宝网商家事业部副总裁王文彬

工具类应用存在巨大市场

记者 / 陈博

淘宝公布的2011中国网购服务市场数据显示，2010年中国网购服务市场规模额为20亿，预计2011年交易额达50亿，同比增速达到了150%，2012年预测服务市场规模将达到125亿，未来三年将迎来井喷式发展。其中工具类软件是卖家使用最多的服务，目前有150万卖家有付费行为，平均订购工具类软件2.75个。淘宝商家事业部副总裁王文彬表示：“面对电商服务市场的巨大契机，在服务平台上创业不再是梦想。”

《程序员》：淘宝目前对第三方开发者的开放态度如何？将开放哪些区域？

王文彬：首先淘宝建立的是一个开放平台，所以我们鼓励第三方进入。当然，淘宝也有一些官方工具，有些是历史原因形成的，是我们必须拥有的基础工具。同时这是一个电子商务开放平台，跟一般的社交网络开放平台有很大不同，它开放的不只是人与人的关系，也开放了商品交易，所以其对安全的要求更高。例如：有一些比较敏感的行业性数据，比如修改价格、修改商品属性等。所以在第三方调用API时，我们会弹出一个窗口，需要卖家确认授权给这个API后才能继续调用。只要是通过卖家授权拿到的数据，我们都会将其权限开放，这是我们开放的原则。

《程序员》：淘宝有哪些措施保证用户数据的安全？

王文彬：对于数据调用，我们需要考虑数据的隐私性、平台的安全性。主要有两大措施：第一是卖家可以授权给第三方工具来调用他自己的数据，但他只能调用自己的数据，不能调用别人的数据。包括消费者的数据，我们也会进行限制，并不是全部都可以下载；第二是对调用次数的限制。因为软件系统可能出现问题，如果对数据全部调用，软件系统可能出现问题，所以我们会确定他今天调用数据的安全性，以及不会危害到淘宝平台的稳定。现在我们也慢慢在做云端的工作，希望用户将一些数据存储在阿里巴巴的云计算平台里，可以更有效地保护企业的安全。

《程序员》：第三方开发者可以在淘宝这个平台上获得哪些特有的发展优势？

王文彬：淘宝从2008年开始做开放平台，现在这些工具类应用的成果都是从开放平台上找出来的。根据数据统计，在2011年9月我们的开放平台平均每分钟帮助1000多个卖家，处理8000个订单，卖家端应用调用API超过100万次，每天API的调用次数为15亿，说明卖家已经离不开这个平台了。同时今年线上的第三方工具订购额达到了8亿，并且是卖家愿意出钱选购这些工具，说明卖家对工具的需求是刚性需求。

年底我们即将推出的新卖家扶持计划中有一些是免费工具，我们的目的是帮助开发者和卖家，卖家可以先试用这些免费工具，在真正有需求时再定制。这可以理解为一个软件的免费版，虽然扶持的是中小卖家，但受益的也是开发者。我们有几十个人的运营团队来分析哪些软件是卖家会受益的，我们也针对开发者、运营团队，给他们提供建议策略。先扶持中小卖家，也是希望培养他们使用软件的忠诚度。

《程序员》：淘宝各类型管理软件需求如何？主要集中于哪些业务上？对第三方开发者的招募条件是什么？

王文彬：主要是一些后台管理、在线订购以及前台营销的软件，工具类的占大多数。比如装修模板、前台营销，后台的进销存、库存管理工具、定单打印工具、用户关系管理工具、客服绩效。比较新生的是数据分析。淘宝对开发者，没有特殊的要求。只要你的产品够好，我们就欢迎。但只有



王文彬认为工具类应用存在巨大市场

在正式上线后，我们才会给你正式调用线上数据的权限。

《程序员》：目前已经与淘宝合作的第三方开发者，他们的生存状态如何？有哪些盈利模式？有什么问题是要特别注意的？

王文彬：根据我们的调查，年收入15万~30万可以支撑1~2人的小型团队，但怎样保证稳定成长是很多创业团队遇到的问题。小型团队有很多成长机会，但要能够持久，能够继续壮大，就需要从团队向企业转型。

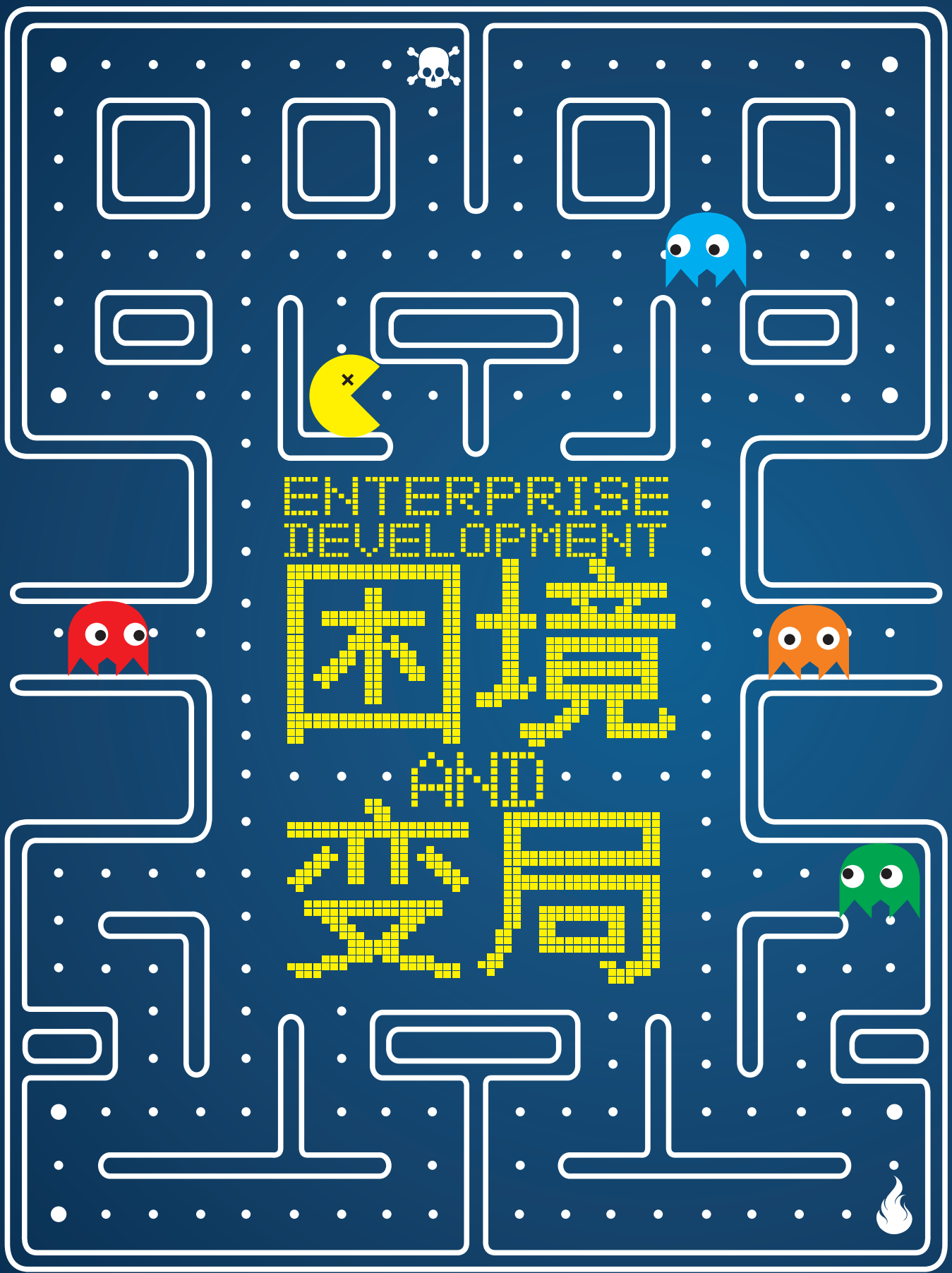
我们能做的就是让开放平台上的竞争公平化、透明化，为开发者提供一个良好的生态环境。但开发者要成长，必须能够抓住卖家的需求，开发出好的产品。市场里一定有人会成长，有人会掉队，淘宝不会去干涉，我们唯一能够做的就是把机制完善。我们与第三方开发者的合作方式是采取收入分成，开发者拿70%，我们30%。并且近两年对于年收入15万以下的新生开发者，我们不参与分成。

《程序员》：现在淘宝上有哪些优秀的第三方开发者产品？请举几个典型的例子。

王文彬：北京有一个叫李大善人，2010年底他个人开发了“收藏有礼”，曾达到一天八万的收入。还有光云软件，他们也是从一个很小的团队开始发展，经过两年时间，年收入已经过千万。大部分淘宝开发软件的团队都是从小做起，包括管易、e店宝这样现在比较大的企业。只是有些成长较快，有些成长较慢。

《程序员》：您对第三方开发者的未来怎么看？有什么寄语？

王文彬：从开发者角度来说，现在整个市场还属于初创期，存在很多机会，但因为大量开发者的进入，竞争也日趋激烈。开发者必须做到：第一，需要了解淘宝的技术框架；第二，有比较完整的产品思维，在产品定位和挖掘卖家需求上，许多开发者还比较欠缺；第三，不要想着一次就开发出高全面性的产品，而是应该先从垂直市场开始，在细分市场有一定的用户群后，再去考虑如何拓展市场。在这个平台上创业，开发者从技术支撑到产品支撑都十分重要，我建议开发团队更多的从产品角度进行思考，这样才能走得更远。


ENTERPRISE
DEVELOPMENT
AND



企业开发的困境与变局

文 / 刘江

算起来，《程序员》已经有几年时间没有大篇幅讨论企业软件开发这个话题了。这其实挺奇怪的。要知道，按类别来分，国内从事企业软件开发的技术人员是最多的，从CSDN和《程序员》联合举办的“2011软件开发者年度调查”结果来看，这个数据超过三分之一（互联网将近30%，嵌入式/工控16.3%）。更让人奇怪的是，企业软件从我们视野里消失，却几乎没有人注意到。

令人尴尬的事情还没有完。几年前我为Martin Fowler的企业软件开发名著《Patterns of Enterprise Application Architecture》做英文注释版时，就注意到企业软件或者企业应用这个业内已经司空见惯、耳熟能详的名词本身，居然没有特别明确的定义。时至今日，这一情况仍然没有改变。最靠谱的维基百科Enterprise Software英文词条里，虽然内容丰满了许多，但还是没有给出令人信服的定义……

共识当然也是有的，企业软件针对的是企事业单位、政府等机构（因为并非只针对企业，所以我习惯用企业级这样的术语，但在实际使用中似乎区别不大）而非面向个人尤其是普通消费者，支撑着机构业务各阶段的运作，支持机构内外人员之间的沟通和协作。

但共识似乎也仅此而已。同一个Gartner网站的术语表里，enterprise applications词条中强调企业内的所有核心业务流程和各个步骤，而enterprise application software中却将办公套件、即时通信、电子邮件甚至内容管理系统等都纳入进来。从本期封面报道中的其他文章也可以看到，即使是资深的业内人士对企业软件的理解也不尽相同。

行业与技术的发展，也在改变许多原有的观念。Martin Fowler对企业应用的定义更强调其数据的复杂和数量，可是现在企业应用的数据从量上来说，已经远远无法与互联网相提并论。有人认为企业应用是相对于互联网和个人软件的，那Salesforce、Yammer等SaaS和PaaS该划为哪一类呢？淘宝、支付宝、去哪儿等面向商家的支付、计费、业务流程等功能，又该算哪一类？

企业软件开发的困境

更尴尬的，是企业软件的现状。

在国内，企业软件中除OA和财务软件外，总体上都是复制国外成功经验发展起来的，原创性本来就不足。在经历了较快增长之后，近年来行业进入了瓶颈期，发展停滞，困扰企业的许多固有老问题一直得不到解决。起步科技总裁马科对本行业有一个很形象的说法：“像装修行业”。他在今年《程序员》9月刊《怎样破解企业管理软件的困局》一文中，列举了企业软件的几大问题：整体混乱，低水平竞争，缺乏足够的质量保证；开发效率低，周期长，应变能力差，往往计划赶不上用户需求变化；缺乏统一的标准，各个开发厂商开发的应用系统往往平台不一，数据格式不一，各自为政，各不相通，无法共享资源；整个行业缺乏创新，技术落后，工具原始，人才流失严重。

八百客副总裁李淼也表示，目前行业处于一种军阀混战、割据的发展阶段，新的在线企业软件领域中大部分厂商也都是在市场上疯狂砸钱，业绩并不理想，而且由于产品同质化严重，一些实力较弱的会被拖垮。

对此，在外企和国企都有过不少从业经历的惠普架构师汪洋感触很多，他认为国内很多所谓的企业软件项目其实都达不到“企业级”的标准，对可靠性、灵活性、重用性、可维护性考虑甚

少，架构粗糙，缺乏必要的可行性和技术风险分析。

鼎基伟业COO杨军也同意整个企业软件市场疲软，他特别指出2011年国内的商业智能市场是生存困难的一年，无论产品厂商还是实施商、甚至咨询商均面临着不小的困难，对市场中从事低端实施的企业更是如此。行业巨头之一的东南融通因“造假案”而摘牌退市，直到公司大规模裁员，濒临绝境，对市场来说是一个沉重的打击。

即使放大到全球，企业软件相关话题也已经被边缘化很久了。

2007年底，影响力很大的技术博客Robert Scoble发表了“为什么企业软件丧失了吸引力”（Why enterprise software isn’t sexy）一文，引起了一场论战。这篇文章的缘起，恰恰是因为盖茨在一次圆桌访谈中对主流媒体只报道消费类互联网而无视企业级软件不满。

最初，来自企业软件圈子的反击似乎很给力，企业软件要关心的就是安全、稳定、可扩展等不吸引人的特性嘛，你Scoble真是个外行。这对在微软、NEC和RackSpace等企业IT领域工作多年的Scoble来说当然很不公平。只要涉及企业软件，他的文章经常能成为媒体焦点，你见过这样的外行吗？

但很快，企业软件用户体验差、需要改进在论战等中渐渐成为双方的共识。以《IT不再重要》和《云计算大变革》等著作闻名的Nicolas Carr撰文指出，真正外行的很可能是自己，用户不喜欢用的产品肯定将面临巨大的颠覆危机。

其实企业软件在互联网开发人员眼中早已经成为笑谈。Ruby on Rails的创造者David Heinemeier Hansson在2005年底就发表过一篇引起热议的文章，其中对2006年做出这样的预测：

> 企业一词将成为软件开发者与用户之间的骂人话。企业软件厂商的成本继续上升，而质量继续下降。这些软件的使用者（他们想要的是简单、轻便、易用的工具）会对软件购买者（他们想要的是臃肿的功能列表）造反。
>
> 这将使企业软件厂商的用户开始流失到更易用、成本更低的解决方案，这些方案可以满足80%所需（剩下的20%根本不值十倍乃至百倍的价格）。
>
> 到2006年底，企业这个词就等同于笨重、昂贵、过时。

Palantir是PayPal创始人Peter Thiel等创办的面向政府和金融企业的信息分析公司，其产品被用于反恐和网络犯罪防范以及经济刺激计划的责任审计。公司的CEO Alex Karp在去年的访谈中表示，2005-2009年间在硅谷你要是创办企业软件公司，就好象是招了一帮工程师在城市中心搞马戏团那么搞笑，要知道那时候“企业”可是一个脏字儿……

除了价高、开发周期长、用户体验等明显弱项之外，传统上企业软件曾经代表着更先进的技术，更好的安全性、稳定性、性能、可伸缩性等的。但时过境迁，现在以互联网为代表的消费应用，在用户和数据量激增的压力下，各项技术也渐渐赶超乃至处于领先（我亲耳听到一位国内大型互联网企业的高级工程师对两家知名国际IT软硬件提供商的资深技术经理说：“你们的产品和解决方案已经根本满足不了我们的需要，我们所要处理的数据量你们连见都没有见过……”）。很明显的趋势是，最新的热点技术包括云计算核心的Hadoop、NoSQL等基本上完全来自互联网。

企业软件落后了。

企业软件的问题何在？

iPhone和iPad在企业世界攻势迅猛的今天，再来看2007年的那场论战，会发现企业应用和消费者应用之间巨大的用户体验的确是问题的要害。当时软件开发殿堂级人物Ed Yourdon发表的观点最经得起时间的考验：企业软件的易用性与安全、可扩展和性能并不是互斥的，Mac上的企业软件就往往比Windows上的好；直接用户的话语权将越来越大，他们会从浏览器、手机开始作出自己的选择，CIO和丑陋难用的企业软件将在博弈中最终输掉。

企业软件落后的原因，究其根本是一个社会问题。Robert Scoble最初文章已经指出：企业软件之所以媒体不太关心，是因为除了CIO外一般人虽然是用户，但根本没有话语权。记

者和博客写文章是为了有更多读者来浏览，然后广告主才会买单，写企业软件？只有几个CIO看啊。

由于产品研发主要围绕企业中负责采购的CIO/经理（可能并不真正懂技术），而非真正的最终用户，所以产品、企业之间的竞争都是畸形的。只要采购经理能交差，产品贵一些不要紧，维护成本高一些不要紧，难用一点不要紧，甚至白花了钱、没啥用处也不要紧。功能全、未来的承诺、buzzword是他们看重的东西。这种中间层的存在，直接导致了对用户体验的忽视，技术落后，流程陈旧，人才流失，进入了恶性循环。

互联网应用之所以有较好的用户体验，是因为能够向数以百万计的用户直接而快速地发布产品，能够快速获得对产品研发至关重要的反馈，同时公司能够保持敏捷和创新的文化，敢于打破陈规。此外，用户容易切换，竞争激烈，也是重要的驱动力。

就在2007年论战之前不久，时任《纽约时报》网站设计总监的Khoi Vinh也发表过一篇文章抨击企业软件缺乏基本的用户体验设计，简直是怪胎，大部分软件中甚至一些非常简单的操作都很难用。

美国第一任CIO Vivek Kundra也曾抱怨在办公室里昂贵的企业软件用得郁闷，而回到家用起Twitter、Facebook就神清气爽，这很可能是他推动Cloud First战略，大力采用许多Web 2.0和SaaS服务的初衷之一。

曾担任Oracle产品营销副总和SAP总经理的Dennis Moore指出，除了产品的用户体验设计外，企业本身不合理的业务流程也是使软件难用的重要原因，而这是企业软件公司很难通过研发和实施解决的。传统企业往往害怕和抗拒变革，改进业务流程阻碍很大。

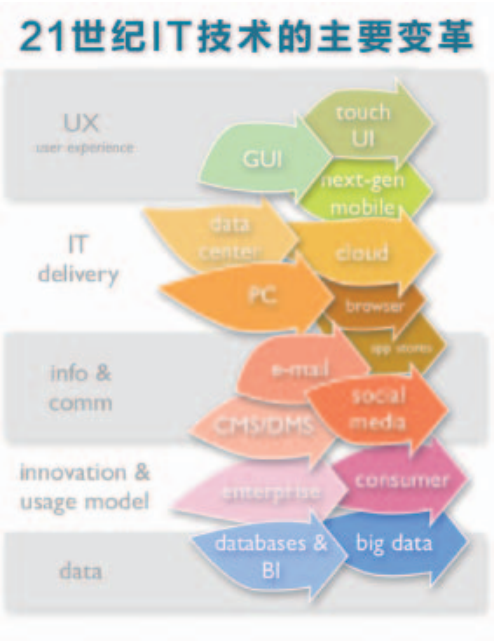
国内的情况更加复杂。由于信息化水平低，很多企业软件应用还处于非常原始的阶段。八百客副总裁李淼透露，国外企业软件厂商的竞争能力很强，它们能只提供一个平台产品，剩下的开发全靠客户自己。而国内客户则不具备这样的能力，需要厂商开发应用给客户提供。他举例说，八百客的客户中98%没有CIO，70%的客户连Excel表还用不好，在这种情况下，要求他们走自己开发的道路，确实非常困难。而惠普的汪洋更指出了国内企业软件企业的诸多问题。公司急功近利，更多的是在赶项目进度，“应用破破烂烂，功能马马虎虎，性能差”的软件项目屡见不鲜。在这种大环境下，技术人员地位往往不高，多数干技术不到10年就开始转型非技术岗位，而国外很多技术人员都是终身制，这意味着我们最多用10年的时间去做一个事情，而别人用一生在做，其中的技术积累和沉淀差别巨大。人员水平良莠不齐，能力和经验缺乏，也使开发流程不完善，敏捷等先进方法不易推行。此外，国内公司总体上规模小、实力弱，也是造成失败案例较多的原因。

而解决这一痼疾，孕育着巨大的商机。种种迹象表明，企业软件正面临涅槃重生。

企业软件的变局

近几年来，兴起了云计算、移动、社会化、海量数据（Big Data）和IT消费化这五大趋势（企业软件专家Dion Hinchcliffe称之为Big Five），对企业软件影响巨大。大变局已经开始。

在国外，Marc Andereessen和Peter Thiel等当今硅谷最有影响的投资人和思想者，都认为企业软件市场大有可为，并投入了大笔资金。



Aaron Levie是面向企业提供云存储和内容管理服务的新锐公司Box.net的创始人，该公司已经拥有众多财富500强客户。他去年9月曾撰文“Enterprise Software Is Sexy Again”，一面回应Scoble三年前的话题，一面宣告新时代的到来。

他发现，2009年以后，许多硅谷创业者和工程师们开始雄心勃勃地进军企业市场，诞生了一批带有互联网基因的新兴企业软件公司。除了市值已达140多亿美元的先驱Salesforce之外，Facebook创始人之一Dustin Moskovitz创办的在线协作平台Asana，已经上市的企业社会化平台Jive，人力资源平台Workday，在线呼叫中心LiveOps，代收费已经超过10亿美元的客户订阅与支付平台Zuora，估值逼近10亿美元的Palantir等等都是其中的代表。

微软、IBM、Oracle、SAP等主导的企业市场相对互联网来说的确是更大的市场，预算非常惊人，高达几千亿美元，仅美国政府每年在IT上的开支就高达800亿。相比之下，Google、Yahoo!、Facebook等众多互联网企业所赖以生存的重要收入来源 ——美国网络广告市场总值仅242亿。

新兴的企业软件公司未必会直接与借传统C/S模式发展起来的企业软件巨头对抗，但云是一种天然民主化的驱动力，消除了来自供应商的所有不自然的渠道、销售和分销优势。有了云，机构的任何用户都是潜在的买主——自下而上的技术采纳正逐渐取代自上而下的部署。

新兴的企业软件公司所要解决的是十年前不存在或者无法解决的问题。今天的企业，有海量的数据要分析和管理，有各属不同角色的员工要共同协作，有分布在全世界的潜在客户要营销，有许多业务问题要实时地解决。他们需要更强大的技术迎接这些挑战。与SharePoint三年才出一个版本，而且少量的补丁都会让IT人员麻烦不已相比，互联网的敏捷研发和发布显然有巨大的优势。

同时，新的企业服务将通过免费增值（Freemium）商业模式以互联网规模运营，与最流行的消费应用一样迅速获得成百万的用户，创业公司也有前所未有的机遇。比如，企业微博Yammer每个季度的收入都在翻番。

新的企业软件公司的文化与消费互联网公司更像。他们不去组建强悍的销售团队，而是专注于产品，将其作为获取用户的最佳方法。按使用和订阅的计费模式取代了传统的许可模式，用户真正使用之后，才付费，这也促使企业软件公司开发用户需要而且真正使用的产品，而不是能卖得更好的产品。

对于国内的情况，起步科技马科认为，现在是中国企业软件行业的机遇期。没有哪个国家这些年里新增了这么多企业，而且以高速在发展，市场环境、商业模式都在发生快速变化，中国企业软件行业有可能创造一些前所未有的变革。但企业软件开发者要抓住这一机遇，必须以开放、积极的心态，拥抱变化，向互联网学习。不仅学习技术，也要学习商业模式。他的观点我非常认同。

八百客李淼将国内企业软件的变化总结为四点：数据集中化；平台化，产品高度集成，可以根据需求随时修改；快捷化，就是越来越傻瓜化，比如移动版、语音识别、菜单的简洁化、功能的简单化等；扁平化，公司不再像日本那样有十几层架构，而会像美国现在普遍采用的四层结构那样，基层和CEO 之间，只隔着两个人，这将大大改进管理，因为不干活的人越来越少了。

传统企业软件公司正面临Clayton Christensen创新者窘境的严峻考验。但SAP、IBM等在自己的企业内部与产品中也有很多企业2.0理念的成功实践。大变局之下，挑战与机遇共存。Box.net公司的经验值得大家借鉴：

1. 让客户感觉自己是公司的一部分——这一点所有企业软件公司都应该向Zappos学习；

2. 收入取决于客户的成功与使用——如果人家不用，你的产品就一文不值；

3. 开发人们每次使用时都会兴奋的产品——这做起来难，忘起来容易；

4. 快速前进，保持朝气，不断创新——自满必然导致优秀人才和客户的流失；

5. 真正解决企业的大问题——问题越大，回报越大。

浅析Java企业开发

文 / 张龙

人们普遍认为与移动互联网开发相比，企业开发所用的技术陈旧、保守，交付周期长，对新技术、新趋势的接受度低，稍显笨重，如何走出人们固有的印象呢？本文做了一些探讨。

与互联网开发相比，企业开发确实存在技术略显陈旧、对高并发、高性能等非功能性需求不太高的事实，但企业开发显然也有自己的特点，如对数据的一致性要求较高、对系统的稳定性有严格界定等。本文将粗浅地分析一些企业开发（特别是Java）的现状，同时展望未来技术的发展趋势。一家之言，欢迎各位同仁一起讨论。

开源框架Struts、Hibernate与MyBatis和Spring

毫无疑问，Struts、Spring、Hibernate与MyBatis在Java企业开发中仍占据极其重要的地位。Spring也从前几年的“高端”框架蜕变成如今的“平民”框架。在几年前，掌握了Spring的Java开发人员会被看作“高端人才”，各大企业也将是否掌握了这几个框架作为面试Java开发人员的标准。但现在这些框架已然成为Java企业开发人员的“标配”。然而，毋庸置疑的是Spring所倡导的控制反转（IoC）与面向切面编程技术（AOP）对软件开发领域所带来的震撼影响。这些理念早已从Java领域延展到了其他语言与技术当中，如.NET领域中的Spring.NET等。下面对这几个框架展开论述。



图1　Struts2实际上是由WebWork2与Struts1合并而来的

Struts

做Java开发的技术人员应该都清楚，Struts是很早就将MVC模式引入到Web开发领域中的框架之一。从Struts1诞生至今已有近10年的光景，即便到现在仍有不少企业在使用Struts1来开发，虽然其中有不少遗留项目，但不得不说Struts1已经深入人心。它迫使开发人员按照MVC模式对项目模块进行切割，这么做自然有利于项目的开发与后期维护。使用MVC模式在开发时会觉得略显繁琐，但从后期维护的角度来看，带来的价值远远胜过开发时多付出代价。虽然在Struts1之后涌现出大量Web MVC框架，如WebWork、Spring MVC、Struts2等，但设计理念基本相似，都是围绕着模型（M）、视图（V）与控制器（C）展开的，因此说Struts1是Web MVC框架鼻祖并不为过。

然而，Struts1的弊端也日益暴露出来，如与Servlet API的紧密耦合，不利于单元测试（虽有诸如Cactus之类的框架可通过继承方式实现对其的单元测试，但仍稍显笨重与繁琐；也可以使用Jetty之类的嵌入式服务器进行容器内测试，但代价高昂）、项目中充斥了大量FormBean、扩展不便；正是此背景下，基于WebWork2的Struts2框架应运而生。虽然Struts1与Struts2仅是版本号不同，实际上两个框架间没有任何必然联系，完全可以认为是截然不同的框架（这里不得不提的是WebWork2所依托的网站OpenSymphony不久前关停了，让人唏嘘不已）。Struts2实际上是完全基于WebWork2的，之所以命名为Struts2完全是当时Struts1太过流行，Struts2想要借助于这个名字让开发者快速接受这个框架而已（参见图1）。Struts2从2007年初诞生到现在走过了近4年的光景，企业对其

的态度也从观望变成欣然接受。相对Struts1，Struts2融合了近几年软件开发领域的一些新思维与模式，更适合现代企业软件的开发特点与流程。与Struts1相比，Struts2的特点如表1所示。

表1 Struts1与Struts2的横向比较

特点	Struts1	Struts2
单元测试友好性	与Servlet API耦合度极高，非常不利于单元测试	与Servlet API解耦，开发者可以很方便地使用JUnit或TestNG等对其进行单元测试，从而脱离了容器的制约
表单与Action的映射	通过FormBean桥接，造成大量FormBean充斥于项目当中	简化了表单与Action之间的值映射，移除了FormBean
插件体系	支持，但不够强大，扩展性差	插件式的体系架构，可以方便地扩展框架；目前Struts2官方网站上已经有几十种插件可供使用
拦截器	不支持	大量应用拦截器，将框架功能划分为一个个的拦截器，便于功能的累加与扩展
标签库	支持，但功能较差	基于OGNL的ValueStack，大大增强了自定义标签的表现力与功能性
Action的创建	单实例Action，多个请求共享同一个Action实例	多实例Action，每个请求对应一个Action实例

但Struts2从诞生到现在也走过不少弯路，首当其冲的是对AJAX的封装。发布伊始，Struts2本身集成了Dojo这一强大的AJAX框架，但在Struts2框架的制约下，Dojo的封装显得不伦不类。Struts2不过是将Dojo的Widget封装成标签库供前端开发人员使用，然而这种封装大大限制了Dojo的能力，很多开发者对此颇为不满。后来，Struts2框架将Dojo的封装剥离出来成为一个插件，但若要在页面中使用Dojo，还是建议使用原生Dojo框架而非Struts2的封装。此外，鉴于jQuery的流行，很多AJAX的功能完全可以由jQuery轻松实现，没必要引入重量级的Dojo框架。Struts2官网的插件列表中也有相应的struts2-jquery插件，但这个插件显得不伦不类。现在，新的Java企业项目一般会选择Struts2而非Struts1，没有历史包袱的束缚，可以最大限度地利用Struts2带来的优秀思想与设计理念。但对一些遗留项目，如SAP项目，Struts1还是不得不用的框架。

Hibernate与MyBatis

Hibernate与MyBatis是目前企业使用最多的对象关系映射框架。尤其是MyBatis，脱离Apache自成门户后发展速度很快（原来的iBatis已经不建议使用，除非是遗留项目，新项目应用MyBatis）。现在的Java企业开发中，大部分项目要么采用Hibernate，要么采用MyBatis。二者间的重要差别在是否使用全自动的对象关系映射：Hibernate为开发者提供一站式服务；而MyBatis则是半自动化的框架，开发者需要手写SQL语句。

使用MyBatis更易于调优和分析数据库（因为某些复杂的SQL语句需要DBA调优，而DBA未必了解Hibernate，只有等到运行时控制台输出SQL语句后才能做进一步分析，这一点MyBatis优势很明显）。很多开发人员对Hibernate的底层机制并不熟悉，以为Hibernate是性能低下的罪魁祸首，殊不知Hibernate是双刃剑，能否用好全在经验积累与对框架的理解程度。

Hibernate的创建者Gavin King曾说过：对于Hibernate来说，最棒的并不是对象关系映射，而是缓存的设计。Hibernate与MyBatis已经占据了Java企业开发中对象关系映射领域的半壁江山。正是Hibernate等对象关系框架的存在才使得JPA横空出世，不得不感叹开源社区影响之大。

Spring

如果哪个Java框架或库没有集成Spring都不好意思和人打招呼，也从侧面印证了Spring在Java企业开发中的江湖地位。Spring凭借其控制反转与面向切面编程颠覆了Java企业开发中非EJB不用的状态，将Java从重量级的旧世界带到了轻量级的新世界，直接导致EJB 3.0出现和EJB 2.0退出舞台。若没有Rod Johnson，现在也许我们还在受EJB 2.0的压迫吧？

Spring倡导的轻量级编程，摆脱了EJB 2.0纷繁复杂的各种接口与异常，让Java企业开发重回正确的轨道上来。近几年Spring生态圈日渐繁荣，在Spring Core基础上衍生了众多的框架，如Spring Security、Spring Integration、Spring Batch、Spring Social、Spring Web Flow、Spring Data，甚至还进军到移动开发领域，如Spring Mobile、Spring Android等，但这些移动框架会有多少开发者使用还是个疑问。

Spring MVC是继Struts之后Java领域最为优秀的MVC Web框架，但由于历史原因，很多企业在技术选型时还会选Struts2。这一方面是技术积累，另一方面是Struts2确实能够满足项目的需求。当然也有不少企业在使用Spring MVC开发，Spring MVC的组件粒度要比Struts2小很多，更易于定制和配置，由于源自Spring，与Spring的集成度更好。

经过多年的发展，Spring生态圈已经非常成熟，但带来的问题是Spring是否会走上EJB的老路。现在Spring变得异常庞大与复杂，尤其是大量XML文件让开发者不堪重负（虽然可以选择Annotation，但还是非常复杂）。从Spring发展趋势看，其目的是提供一站式开发平台（已提供了Web服务器tc Server），但在可预见的未来，Spring依旧是Java企业开发领域当之无愧的王者，也是众多企业首选开发平台。

综上所述，目前的Java企业开发中，开源框架Struts、Spring、Hibernate与MyBatis是不可或缺的中坚力量，很多公司在这上面积累了大量经验，同时这方面人才招聘也较容易。

Flex

近年来，Flex在Java企业开发中大放异彩，这得益于Adobe不遗余力的推广与发展，也是因为现代企业开发中对于用户界面的各种需求，尤其是复杂报表的需求所导致。

在RIA系统中基本上有两种技术选型：一是JavaScript+CSS，二是Flex、Silverlight等富客户端技术。前者的代表性作品有ExtJS、Dojo、jQuery及相关插件等JavaScript框架（库），但由于开发成本、学习曲线以及客户端浏览器（很多企业客户现在依然在使用IE7，甚至IE6）等限制因素，Flex成为了企业开发的救命稻草。Flex可以很轻松地实现与后端Java系统的通信（基于HTTPService、Web Service以及Remote Object），大大降低了Java开发人员的学习成本；Flex富于表现力的UI也深受企业客户的青睐。虽然前不久Adobe宣布放弃移动版Flash开发并打算将Flex捐献给Apache软件基金会，但相信对Flex，Adobe还会加大投资力度。即便Flex被捐献给Apache软件基金会，Adobe开发人员也会全程参与其开发与管理。

由于Flex编译后生成的swf文件过于庞大，对带宽受限的用户会成为一个很严重的问题。同样的功能，使用HTML+JavaScript+CSS生成的文件只需几十KB，但Flex生成的swf文件很容易就达到几百KB，这也是开发中不得不考虑的现实问题。当前国内的网络状况下，互联网整站Flex开发还不太现实；但Intranet下的整站Flex则没有任何问题。Flex也将保持企业开发中的重要角色。

> jBPM使用了私有的标记语言jPDL，而Activiti则支持BPMN 2.0标准。众所周知，标准的可扩展性远远好于私有标记语言，从这一点来看，Activiti发展潜力是巨大的。

OSGi

OSGi可谓Java模块化开发的先驱，Eclipse的底层架构现已完全转向OSGi，甚至Oracle的JDeveloper也已经将OSGi作为底层基础设施。但OSGi在目前国内不温不火，虽有企业倡导OSGi的模块化开发模式，跟从者依然门可罗雀，OSGi在国内还有很长的路要走。理念是先进的，但理想与现实还有不小的差距。另外不得不提的因素是OSGi相关开发人员稀缺，这也是制约着OSGi在国内发展的瓶颈。

工作流

在国内目前的企业开发中，工作流可谓“标配”。Java工作流领域中，jBPM依然是无可争议的王者。大量企业系统的工作流模块都在使用jBPM框架，因为它最久远，各个公司积累的经验很多，但这里想说的并不是jBPM，而是Activiti。

众所周知，jBPM的创建者Tom Baeyens对jBPM未来发展方向和技术架构持不同想法，于是离开了JBoss并加入Alfresco，带领一群开发者创立了Activiti。jBPM和Activiti都出自Tom

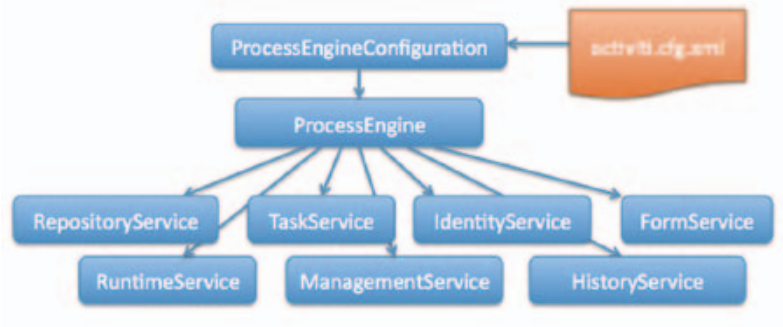


图2 Activiti架构图

Baeyens之手，因此这两个框架从学习角度来看有很多相似之处。而jBPM在Tom Baeyens离开后似乎陷入了停滞状态，新发布的jBPM 5完全摒弃了jBPM 4，而基于Drools Flow重新开发。使用过jBPM的开发者都清楚，jBPM 4与jBPM 3之间有明显差异，几乎无法向后兼容，而jBPM 5与jBPM 4又有天壤之别，导致众多用户无法升级。因此国内还有很多jBPM用户使用jBPM 3，并非不想升级，而是根本无法升级，升级而重新开发的代价未免太大。

反观Activiti，在jBPM 4的基础之上实现了更为强大的功能，包括Core Engine、Activiti Explorer、Activiti Cycle、Activiti Probe、集成Mule、集成Spring等。Activiti保持着每月发布一个小版本的势头，并且向后兼容，对开发者非常友好。由于与jBPM同出自Tom Baeyens之手，其实现架构对于熟悉jBPM的开发者来说会感到非常亲切，如图2所示。

对熟悉jBPM的开发者来说，若2图中橘黄色块的文件名不是activiti.cfg.xml，而是jbpm.cfg.xml，那一定会以为是jBPM的架构图。然而这正是Activiti的架构图，与jBPM一样使用了Façade模式，通过ProcessEngine流程引擎公开各种Service，而底层各种Service则用于完成不同功能，如部署流程的RepositoryService、用于处理任务的TaskService、用于标识用户及组信息的IdentityService等。

与jBPM5相比，Activiti最大特点是保留了PVM（Process Virtual Machine，流程虚拟机），因此推荐使用jBPM4的用户转向Activiti而非jBPM 5。jBPM使用了私有的标记语言jPDL，而Activiti则支持BPMN 2.0标准。众所周知，标准的可扩展性远远好于私有标记语言，从这一点来看，Activiti发展潜力是巨大的。

但目前使用Activiti的成熟案例是少之又少，很多企业还在持观望态度，同时Activiti相关书籍与资料还不丰富。除Manning出版社尚未出版的《Activiti in Action》一书外，市场上还没有Activiti图书，这也是制约着Activiti进一步发展的瓶颈，开发者只能通过Activiti官方网站的资源和邮件列表寻求帮助。但凭借Activiti的快速发展与庞大的开发团队，在未来几年中，相信Activiti一定能力压jBPM成为Java工作流领域的王者。

数据库

提到数据库，企业级开发者脑海中会条件反射出如DB2、Oracle、SQL Server等传统关系型数据库，这是构建企业应用的基石。但近年来随着Facebook、Twitter、Amazon及Google等互联网公司的不断推进，NoSQL如火如荼地发展起来。首先是Google发布BigTable架构的论文，然后是Amazon发布的Dynamo论文。此后，NoSQL产品如雨后春笋般发展起来，代表作有MongoDB、CouchDB、Redis、Cassandra等，每种产品都面向特定领域。

为何会出现NoSQL呢？这主要得益于Web 2.0的蓬勃发展。各种Web 2.0网站在面对高可靠性、高性能等非功能性需求时，传统的关系型数据库已然无法满足业务需求，这时NoSQL就应运而生了。企业应用所用的传统关系型数据库历史悠久、经过了无数的考验，具有极强的稳定性与生命力，同时以SQL这一标准语言作为数据库操纵语言也深得人心。

但在互联网领域，随着网站用户访问量的不断增加，性能问题成为了制约业务发展的瓶颈。这时，单靠数据库的分表分库也已经无法满足高并发、高性能的需求。关系数据库虽然强大，但在面对这种场景时也是无能为力，这时出现的NoSQL正是该问题的解决方案。

NoSQL具有非常灵活的数据模型，与传统的关系型数据库不同（需要建立好表、字段并设定每个字段的数据类型），NoSQL则可以非常方便地存储自定义的数据类型。在Web 2.0应

用中，随着业务的发展，我们很有可能会为表增加或删除字段（主要还是增加，删除的几率非常小）。如果是关系数据库，那么实现起来会非常麻烦，尤其是表中数据量较大的情况下更是如此，但对于NoSQL来说则完全不存在任何问题，因为其数据模型是极其灵活的，只需将新数据插入进来即可。

此外，NoSQL的读写速度也非常快，并不需要保证数据的ACID特性；此外，Web 2.0应用毫无例外地会使用到缓存，对于传统的关系数据库，每次对表的修改都需要更新整个缓存，这是一种代价极其高昂的操作。而对于NoSQL来说，由于缓存是记录级别的，因此每次更新缓存都是一种轻量级的操作，代价也非常低，因此非常适合于互联网应用。

目前，NoSQL在企业开发中的应用并不算多。这主要是因为企业应用所面对的高并发、高性能场景并不如互联网应用那么频繁；此外，企业应用中的数据要求更高的一致性，这一点与互联网应用的差别非常明显。最后，由于很多企业应用都是遗留应用，原系统已经使用了诸如DB2、Oracle之类的关系数据库，若想转到NoSQL也是非常困难的事情。

展望

本文讨论了目前Java企业开发领域的一些现状。从文中可以看出，企业开发还是具有其特点的，诸如要求数据的一致性、系统的稳定性、技术架构的成熟性等。通常来说，一项新技术出现后，首先采用的都是互联网应用，而企业应用只在经历了足够长的时间验证后才会采用，从这一点也可以看出很多企业应用在技术选型问题上所持有的谨慎态度。

但不可否认的是，移动开发正在悄悄步入企业开发领域。至少现在就有不少企业将系统中的很多核心功能抽取出来作为移动客户端，典型的就有工作流系统。这样，系统使用人员就可以通过手持设备轻松完成流程的创建与批复，这也将是未来一段时间企业开发的一个新方向。

至于现在非常火爆的HTML5与CSS3，虽然已经在互联网与移动领域得到了长足的发展，也取得了不俗的成绩，但我认为距离进入企业开发领域还有相当长的一段时间。因为很多企业客户，特别是国内的企业客户依然在使用上个世纪的IE6，这将是制约HTML5与CSS3进入企业开发领域的一个巨大的绊脚石。

基于开源框架Struts、Spring、Hibernate与MyBatis的技术依旧会在未来的一段时间内占据主力地位，而Flex则会越来越多地为企业项目所采纳；至于OSGi则还有很漫长的一段路要走。

总结本文，我认为在Java企业开发中，基于开源框架Struts、Spring、Hibernate与MyBatis的技术依旧会在未来的一段时间内占据主力地位，而Flex则会越来越多地为企业项目所采纳；至于OSGi则还有很漫长的一段路要走。在工作流领域，jBPM还会继续辉煌一段时间，之后则会被Activiti所取代，而NoSQL距离进军企业应用也不是一朝一夕能够实现的。相反，移动开发与企业开发的整合将是未来一段时间的热点，也将是企业开发的一个新方向。



张龙

同济大学软件工程硕士，混迹于外企多年，有一年的创业经历。对Java轻量级框架有一定研究，工作流领域研究与实践者。
新浪微博：http://t.sina.com.cn/fengzhongye。

.NET企业级开发

文 / 汪洋

.NET平台是一个成熟、便捷的平台。特别是在实现“企业级”原则时，可以提供更多的便利。作者点评了.NET企业级开发的若干技术，并且论述了.NET企业级开发的热点话题与前景。

.NET 企业级开发

.NET是一个成熟、便捷的平台。特别是在实现“企业级”原则时，可以提供更多的便利。因为很多朋友对.NET都比较熟悉，下面就首先简要点评一下常用的技术。

WCF的企业级开发

WCF提供了比较简单的编程模型，屏蔽了很多没有必要知道的复杂性，使得开发更加容易。这里主要关注，WCF如何在项目中实现上述企业级的四个原则。

■ **可靠性**：WCF为项目提供的可靠性，主要体现在两个方面：会话的可靠性与通信的安全性。

在WCF中，可以通过事务机制，将一组操作单元（由一个或者多个操作构成）作为整体执行，这一点和数据库中的事务机制的作用类似。WCF也采用了队列的思想，来确保即使在网络失败或发送方和接收方之间的通信崩溃时，依然可以通信。

另外，WCF通过三个部分来实现通信的安全性：传输控制、访问授权控制以及审核机制。传输控制主要是负责提供消息的完整性、机密性和身份验证。访问授权控制主要是通过提供不同授权机制来访问指定的资源和功能。

■ **灵活性**：为了达到这一点，沿用微软一贯的设计风格：管道插件。可以说WCF的每个部分都可以根据需要来自定义功能，例如消息结构、通信协议等。并且在很多时候，可以在不改动源代码的情况，只需要修改配置文件，就改变相关功能。

■ **重用性**：体现在两个方面 ：首先，对原有技术的重用，例如，在实现可靠消息通信时，底层机制采用了MSMQ，在身份验证时，采用标识验证和基于ASP.NET Membership的验证等；其次，使用WCF设计的功能，例如自定义的绑定、消息通信机制，可以在其他项目需要时，直接拿过去使用，因为它们的设计都遵循统一的接口规范。

■ **可维护性**：这一点主要建立在设计的灵活性以及统一性上面。例如，可以将服务契约放在接口上，从而替换不同的实现，这一点比较简单。另外，WCF强大的配置功能，可以实现基本插件式编程。

WPF的企业级开发

每每谈到WPF，似乎就要提及MVVM。但在使用WPF开发时，根据实际情况，不见得非得要用MVVM，就像在使用Winform和ASP.NET Form开发中，不见得非得要用MVP模式一样。考虑使用成本和效率的同时，也要考虑技术人员的能力。下面还是从四个角度阐述WPF的企业级开发。

■ **可靠性**：.NET平台的Winform，是比较成熟、稳定的开发框架，而WPF继承了Winform的优点。不仅如此，WPF 4还向WPF开发人员呈现了若干个关键的Windows 7 Shell新特性。这些Shell特性能促成更丰富的集成用户体验。例如，新的任务栏没那么拥挤了，一眼看去可以传达更多信息。跳转列表给应用程序提供了对与上下文相关的启动任务和文件的访问。

■ **灵活性**：依赖属性机制的提出，为WPF开发提供了非常灵活的选择。同时，强大的数据绑定机制，也使得开发分工有了很大的灵活性——开发人员与界面人员可以单独工作了。

■ **重用性**：WPF在这一点上面做得相当好。在WPF中，可以将所有东西定义为资源（不仅仅是布局信息，也可以将数据等定义为资源），从而在项目中重用。

■ **可维护性**：WPF已经实现了基本的维护思想，例如UI的XAML布局代码与后台的功能代码相分离，并通过命令机制解耦合。为了更好地实现可维护性，还需要做更多的事情，例如使用MVVM模式来解耦合。

下面简要看看WPF中的Command。WPF中的Command机制就是GOF设计模式中Command模式的一种实现，Command模式的UML结构如图1所示。

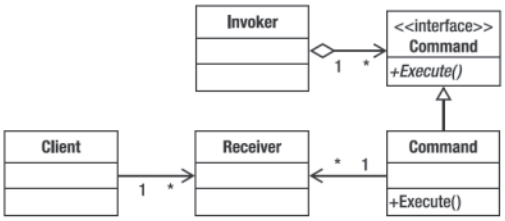


图1 Command模式UML结构图

在.NET中，如果说事件机制和事件的触发者是一对一的关系，那么Command和命令的触发者就是一对多的关系。在采用了Command机制之后，使得很多动作都去执行同一个Command，这就使得命令的发送者和执行者完全分离，同时在更大程度上实现了代码重用，为维护和开发带来了效率。

在WPF中，ICommandSource接受充当了命令执行者（Invoker）的角色，定义如图2所示。



图2 ICommandSource类图

定义代码如下所示：

```
public interface ICommandSource
{
     ICommand Command { get; set; }
       object CommandParameter { get;
set; }
      IInputElement CommandTarget { get;
set; }
}
```

从上面代码可以看出，这个接口中只定义了一个Command。在WPF/Silverlight中从ButtonBase继承的控件（如Button，RadioButton等）实现了这个接口。WPF/Silverlight的框架限制了让控件有多个Command的可能。很多时候，一个Command不能涵盖所有情况，所以在开发中，常借助事件机制来达到想要的结果。

ICommand的结构如图3所示。



图3 ICommand的结构

定义代码如下所示：

```
public interface ICommand
{
event EventHandler
CanExecuteChanged;
bool CanExecute(object parameter);
void Execute(object parameter);
 }
```

对于以上述代码。

■ CanExecute方法根据传入的参数决定是否执行Execute方法，并且触发CanExecuteChanged事件。

■ 当CanExecute方法返回true时，执行Execute方法。

下面看看在WPF中Command机制是如何实现的，如图4所示。

可以看到，WPF中的RoutedCommand和RoutedUICommand实现了ICommand接口，对于图4我们有如下解释。

■ 当按钮被点击时，因为按钮实现了ICommandSource接口，所以此时的按钮便是命令的调用者。按钮执行Command命令（此时具体的ICommand接口的实现者是RoutedCommand）。

■ RoutedCommand对象在自身的Execute方法中调用CommandManager的Executed方法，此方法会遍历程序中所有的WPF元素，以找到合适的CommandBinding。CommandBinding将一个Command命令和一个处理程序联系在一起，例如CommandA命令执行 OperateA方法，

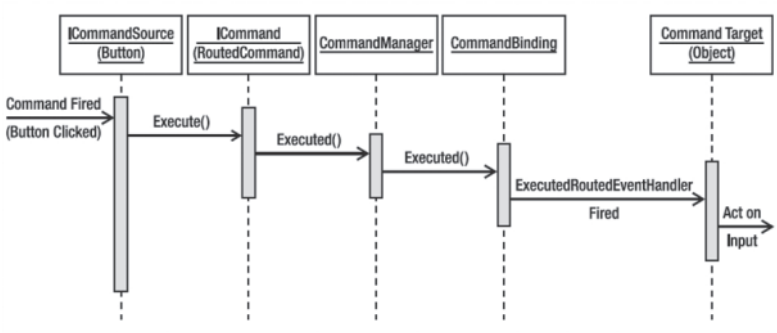


图4 Command机制实现

CommandB命令执行OperateB方法。

可以将此与前端控制器中的Command注册表进行类比。图5展示了CommandBinding、Command以及处理程序之间的关系。

ASP.NET企业级开发

在ASP.NET平台上，微软提供了两个应用开发框架：ASP.NET Webform与ASP.NET MVC，它们是并存的关系。

采用ASP.NET平台来开发，可以轻松实现企业级的四个原则，已有大量资料书籍讲了这个问题，这里再赘述。

我们的关注点常常在性能上。就ASP.NET平台本身而言，是完全支持大并发、提供高性能的。很多时候，一个ASP.NET站点的性能不佳，不是因为ASP.NET天生能力不足，而是设计者和开发者的技术有待提高。

企业级开发的关注点

微软近期的重点可能在移动上面，所以很多新技术也是针对移动开发的。但从之前的叙述可以知道，“企业级开发”完全涵盖移动开发。例如，采用Windows Phone 7开发客户端以及采用WCF技术在服务端提供服务。

此外微软Windows Azure也是以后发展的重点，通过它，可以为企业节省硬件的购买和维护的成本，并提供更多的性能和容错保障。

现在还少有企业会真正将所有应用和数据转移到Windows Azure上，很大原因就是出于数据安全和保密性的考虑。但据我所知，现在已经有很多互联网创业公司和个人采用Windows Azure，这将是一种趋势：充分利用现有成熟的方案和资源，节省成本，使得利润最大化。

.NET企业开发总结

.NET平台的好处很明显，不足之处也显而易见。最大问题就是，开源技术和框架没有Java平台多，例如，现在流行的NoSQL及其相关框架很多都是采用Java开发的。但幸好还有很多.NET社区不断跟进这件事情，研发自己的开源框架，又有相当多的.NET 驱动组件在使用这些开源框架。

所以对于那些已经或者准备学习.NET的朋友来说，.NET还是很有前途的。我之前也做过Java，后来才转到.NET开发。

我觉得能把.NET平台做熟、做精，以后再跨平台做产品，也是很有帮助的，并且很多原理都是相通的。

谈完前面内容，基本上已经对“企业级”的含义有了比较清楚的认识，同时也清楚了企业级的关注点：可靠性、灵活性、重用性、可维护性。可以说，上面四点是衡量项目是否是符合“企业级”开发的基本标准，但这是一个非技术性的标准。项目最终是否成功，还是以要是否实现客户需求，开发出实用的、带来价值的软件为标准。

所以“开发出满足客户需要的软件”才是“企业级”开发最重要的关注点。为了实现这个关注点，很多人提出了很多开发方式和思想，例如持续交付、重构、精益和Scrum等。

每个行业都有需要完善的地方，软件行业也是一样的。正因如此，我们才坚持在IT路上奋斗，一步步向“企业级”迈进。所以真正实现“企业级开发”任重而道远。



汪洋

惠普架构师、信息分析师。著有《.NET应用架构设计：模式、原则与实践》。曾任上海益思研发管理咨询有限公司首席软件架构专家、软件咨询组副组长。

SOA在当今中国企业的发展现状

文 / 马国耀

SOA虽然不再是Buzz Word，但这也从侧面表明，对它的理解正在深入，使用正在趋于冷静而理性，在历练中得到了长足的发展。但由于SOA自身的局限性和对其使用的质量参差不齐，SOA在当今中国企业的发展现状还需要我们继续讨论。

相信今天已经没有人对SOA的概念感到陌生，或者说没有听到过这个名字。但，你眼中的SOA是什么？1000个人可能会有1000种理解。我认为造成这种理解上的差异，原因有两点。其一，给SOA下定义的组织机构太多，没有统一的标准。若干标准化组织、大厂商等都给出了自己对SOA的理解，但这些理解并不统一。其二，人们对SOA的理解曾一度被解决方案厂商所误导。或者由于短视，或者迫于抢占市场的压力，SOA解决方案供应商们关心的是打着SOA的旗号卖产品。但凡能沾点边儿的，都贴上了SOA的标签。

然而，值得庆幸的是，不论人们是否深入理解SOA，似乎都接受了它，也能大致说出它的几大特点和优势——应用程序服务化、松耦合、灵活的架构、支持敏捷等。

SOA是什么？如何实施SOA？解决这一类问题的文章在出版物和网络上可以搜索到一堆，我就不再重复这些工作了。本文的重点是介绍SOA在当今中国企业中的发展现状。但为了全文结构的完整性，将使用一小节简述我对SOA的理解。

SOA简述

SOA的全称是Service Oriented Architecture，即面向服务的架构。从其名字上看，它有两个核心：一是服务，二是架构。在此基础上，我对SOA是这样理解的。

■ 它以服务为核心。传统的整合架构都是以应用为核心，而SOA里谈的一切都是从服务展开的。

■ 它不是某种特定技术，而是一种架构风格、架构思想或一组指导架构设计原则。在特定风格、思想和原则的指导下设计出来的企业应用架构就是具有SOA的特性和优势的架构。

SOA的目标是实现IT和业务对齐（如图1所示）。这一目标可作为衡量SOA实施成败与否的重要尺度之一。

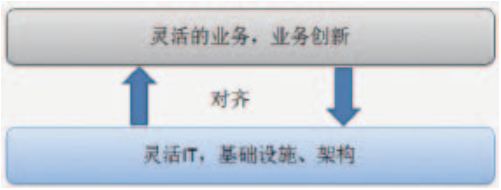


图1 SOA的目标：IT与业务对齐

众所周知，业务是不断发展、变化的，业务创新越来越成为企业核心竞争力的一部分。传统IT支撑业务发展的方式是为新业务构建新的业务系统，设计新的数据库，系统多是“竖井状”的孤岛形式，系统间连接常见的是点对点的缺乏规划的连接方式。久而久之，这样的方式已经无法满足业务发展对IT带来的要求，IT成为企业的严重短板。SOA出现的主要目的就是彻底改变这种局面，使IT变得灵活，与业务对齐，不再成为企业的后腿。甚至能够促进业务的发展，并创造新的业务价值。

为什么SOA能够实现IT和业务对齐呢？这得归功于SOA架构中的若干实施原则。

■ 应用功能的服务化

将应用程序打散成服务是SOA实施的关键

之一。一方面，遗留系统的功能和数据不能丢弃，只有重用才能利用其价值。另一方面，将应用作为整体，是从技术和实施上都非常难于复用的。所以，SOA的实施原则要求将应用中的功能模块抽象成开放、标准、粒度适中的服务，进而促进现有功能重用、避免重复投资、加速新业务开发的速度。

■架构的松耦合原则

SOA的另一重要原则是松耦合，松耦合不仅表现在服务本身，还表现在架构层面。服务的松耦合表现在服务的描述与其底层实现技术是分离的。换言之，一个服务，其底层实现可能是Java，也可能是.NET；架构的松耦合表现在服务消费者与服务提供者的分离，在服务消费者及提供者之间置入仲裁（Mediation）模块。这样，服务提供者对服务的修改对服务消费者产生的影响可降至最低。

■面向服务的整合

面向服务的整合是SOA的核心之一，主要由企业服务总线（ESB）提供这一部分的能力。服务消费者之间既有面向服务的调用，也有面向消息的处理。企业服务总线提供的协议转换、数据格式转化、多种消息处理机制、审计、日志等功能能够最大限度地提高SOA架构的灵活性和扩展性。

■服务的组合及编排

服务梳理出来之后，基于服务的组合和编排就成为可能。服务的组合和编排能够最大限度地重用现有功能，支持业务的创新。偏重于技术层面的服务/消息组合可由ESB提供，而偏业务层面的服务组合和编排可通过BPM组件实现。当然，SOA实施原则还有许多，如服务治理、安全访问、信息服务、门户服务的实施原则，本文不再赘述。

中国式SOA在企业中的发展现状

前几年，硝烟弥漫、战火纷飞，各路厂商奋力拼杀在SOA市场上。大厂商拼的是在市场上的统治地位，小厂商若能分得一杯羹也满意了。喧嚣之后总要归于沉静。转眼间，云计算铺天盖地席卷而来，绚丽的云彩抓住了大众的眼球。SOA像是退居幕后的明星、落幕的英雄，很少再被人提起，甚至有人宣布“SOA已死”。那么，SOA真的销声匿迹了吗？中国式SOA的当前状态是怎样的呢？



图2 Gartner技术成熟度模型

SOA呼声渐弱，实际上它越来越火

的确，现在大厂商们都已经不再把SOA作为市场宣传的重点了。但是，SOA却走到了应用最广泛的时期。若对应到Gartner技术成熟度曲线模型，中国当前SOA所处的阶段是图2中的第4个阶段——复苏期。

我们大致可以如此推算SOA在中国的几个阶段，2006年之前是技术萌芽；2006-2008年是过热期；2009年度过了幻灭期；从2010年开始进入复苏期，当前正由复苏期迈向成熟期。

经过几年的历练，中国的SOA不论理论、实践，还是产品都得到了长足的发展。一方面，中国培养了一大批掌握SOA精髓的系统集成商和独立软件提供商，如华为、东软、用友、金蝶、中软……他们或将SOA的理念植入到其解决方案中，或将SOA理念融合进其产品之中。SOA架构师、整合架构师的队伍也不断壮大。另一方面，中国也拥有了自主研发的SOA中间件产品，如普元、炎黄盈动的流程引擎；锐易特、东方通、金蝶的ESB都是市场上的佼佼者。伴随着国人对SOA的理解越来越深入，越来越成熟和专业的SOA产品及解决方案将会随之而来。

SOA已经深入人心。现在，几乎所有待建的系统和平台都要求基于SOA架构。SOA已经成为基本的架构原则，不论是系统间的整合，还是平台的建设，人们会不约而同地采用SOA原则来架构灵活、易于扩展的系统。

中国式SOA的几点不足

尽管SOA已经深入人心，但并非所有的人都能将SOA原则使用得很好。若不能很好运用SOA，就无法收获相应的价值。下面给出几点常见的不足之处，我无法在这篇文章中展开讨论这些问题，但还是期待从业人员正视这些不足之处，找到正确的实施方法。

■ 对SOA的理解参差不齐。若不能充分理解SOA，未能理解SOA的目标和架构原则，就无法正确实施SOA。在一些项目实施过程中，旧系统被直接推翻，而不是先将遗留系统服务化，然后再循序渐进式地改进。若所有旧系统全部推翻后重建，势必会耗费大量时间和人物力才能完成，不符合SOA多次少量投入的特点。此外，许多人理解的服务就是Web Service规范的服务。而事实上，SOA不限定采用何种技术，只要是开放、标准的技术即可，比如JMS、RESTful Service等。

■ 忽略服务梳理和服务描述的重要性。我清楚地记得一个项目，用户通过这样的方式生产Web服务：将系统里的某个通用Java接口中的所有方法全部暴露出来，使用某个WSDL自动生产工具生成一个超级大的WSDL文件以及许多关联的数据描述（XSD）文件。WSDL中暴露了大量原有程序中的Java包结构和数据结构信息。这个例子是一个糟糕的实践。

哪些功能应该封装成服务、服务该如何描述，这些工作是需要规划的。我们需要根据业务、现有系统等方面，全方位（或局部）梳理业务，才能找到那些适合于暴露成服务的功能。确定好应该暴露的服务之后，进一步确定服务规约，描述其输入和输出、与其他系统的依赖关系、服务非功能性需求等。

■ 重视工作流（Workflow），忽视服务自动编排（Service Choreography）。我见到的大多数BPM项目都是关于工作流的，工作流是以审批为主的业务流程，在中国尤为突出。这种流程在某种程度上比此前从一个办公室走到另一办公室、然后再到下一个办公室的场景已经先进许多了。但我们知道除了工作流之外，BPM还有其他形式，比如服务自动编排，这样的场景却在很大程度上被忽视了。

■ 服务治理未得到充分重视。服务治理或管理对于长期SOA实施的成败至关重要。随着SOA的持续发展，企业中的服务会越来越多，服务的变更、服务位置的变化和服务的版本升级相关工作越来越频繁，如何把这些变化造成的影响控制到最小？再者，当服务多了之后，如何快速便捷地找到所需的服务也是极为重要的工作。所以，服务管理和治理越发显得重要。

企业级应用，SOA还是ROA？

随着Web2.0和云计算的兴起，REST应用再次变得火热起来。与REST的相关的词汇也出现了好几个，RESTful Service和ROA （Resources Oriented Architecture）就是典型的代表。甚至有许多人认为ROA会替代SOA。事实上是这样的吗？

认为ROA胜过SOA的人一般指的是RESTful Service胜过SOAP，这种观点产生的原因是RESTful Service简单、扩展性高、高性能（缓存机制）；而大多数SOAP缺省基于RPC方式【注：SOAP并不限制一定使用RPC】，这种机制需要在客户端建立一个代理才模拟服务短的接口，进而调用服务端的服务。相对而言，SOAP的客户端和服务端的耦合要紧密一些，这限制了分布式应用的扩展。另外，RESTful Service推荐的消息传输格式是JSON，它比SOAP要求的数据格式SOAP信封要简单许多，这也是RESTful Service胜出的另一方面。

我认为要回答这个问题，首先需要看看企业级应用的特点以及两种架构风格的适用性。由于遗留原因，目前企业内的大多数应用并非基于Web，应用系统间的交互大多基于RPC方式，ROA很难在这种环境下开枝散叶。

此外，企业应用间互相访问的功能多数是基于过程而非基于资源交互，例如“核准保单”，虽然可以通过REST实现这种场景，但毕竟没有RPC使用起来那么直观。可喜的是，近年来Web应用越来越多地在企业内使用，这为ROA提供了有利的土壤，使得ROA在企业架构中的应用变得多起来。从两种架构风格的适用性角度看，一方面，SOA适合于企业级应用

自不必说，这么多年来SOA一直在企业范围内应用；另一方面，到目前为止，ROA则更多地应用于基于Web的前端应用的数据聚合，所以ROA在企业内的适用性要窄一些。

基于以上考虑，我认为ROA可适用于企业内的Web应用之间的整合和协作。ROA通过RESTful Service的形式暴露服务，这些服务可通过SOA架构进一步与企业的其他非Web应用进行整合；SOA则是更高层次的企业级架构风格，RESTful Service可作为描述服务的开放、标准的形式之一，但不是唯一形式。随着企业级应用越来越Web化，遗留系统正逐步淘汰，ROA可能会成为未来企业级解决方案的重要组成部分。

简言之，SOA不会被替代，ROA的地位会越来越高；二者并非排他关系，而是相辅相成，共荣共赢的关系。

云计算冲击下的SOA

云计算是新宠，就像当年的SOA一样。当市场在大肆宣传云计算时，SOA在一旁淡然地笑了。喜欢挑起争端的人会说：“云计算来了，SOA该靠边站。”这种说法只能当作不负责任的市场噱头，不能当真。然而，二者并非同一层面的事物，不具有可比性，所以也就不存在谁替代谁的问题。相反，云计算和SOA还能彼此促进对方的发展。表现在以下几个方面。

一方面，如果一个企业成功地实施了SOA，当云中某个服务能够为企业带来更大价值时，其架构就能很方便地扩展，将云中服务接入现有架构之中。另一方面，由于业务扩展造成现有服务无法支撑剧增的业务量时，在良好架构的技术上，就能充分利用云计算带来的弹性伸缩和按需功能等方面的优势。

云计算供应商需要以SOA为基础。云计算的核心也是服务，计算、存储、网络、软件等，一切皆服务。那么，如何将这些能力封装成服务，以良定义的接口为外界提供服务呢？这就需要SOA的支撑。所以，云计算的兴起更加促进了SOA的应用，我看到过某些企业使用SOA架构构建SaaS、PaaS应用并对外提供服务。



马国耀

现就职于IBM北亚太战略合作部，为合作伙伴提供产品培训、技术支持和架构咨询等服务。他曾获得吉林大学学士学位和北京大学硕士学位。译著有《云计算与SOA》。

寻找机遇 创造未来
CSDN 热门职位全新推荐

UFIDA 用友

用友软件

诚聘以下职位：

一、Java开发工程师

岗位要求：

1. 两年以上Java开发经验；
2. 熟悉SQL Server或Oracle数据库；
3. 熟悉Eclipse开发工具；
4. 熟练掌握SQL语句，能够熟练进行数据库设计。

二、.NET开发工程师

岗位要求：

1. 两年以上.NET开发经验，具备一定分析设计能力；
2. 良好的C#代码实现能力，熟悉ASP.NET开发技术；
3. 掌握面向对象的设计方法和设计原则。

简历投递：rdzhaopin1@ufida.com.cn

DeNA China

上海纵游网络技术有限公司

我们正在全力招聘中国人才，打造一个全新的中国团队，致力于中国智能手机游戏平台的构筑和运营。愿意加入我们，一起构筑世界著名的智能手机游戏平台吗？

Android/iPhone/Web application，更多职位招聘信息请登录公司主页进行查询

公司主页：www.denachina.com

请以“职位+姓名+CSDN”作为应聘邮件标题，将中英文或中日文简历发至
hr@denachina.com

浅析工作流平台

文 / 徐会生

介绍与实例

“随手画一张流程图，配置好每个步骤执行的操作和负责人，点击一下发布，流程就可以在系统中运行了”。上面这段宏伟的蓝图就是十几年来工作流平台的研发工程师们为之奋斗的理想。

工作流平台的目标是使用更形象的方式来描述流程中的各个步骤和步骤间的关系。相对于文字性的描述，人们更容易理解使用图形的表述方式，大量采用图形来对业务流程进行抽象也就成为工作流平台与其他解决方案的最大区别。

以一个简单的请假流程为例，采用文字性的描述，就会得到如下结果：

- 员工填写请假申请，提交给经理审批；
- 如果经理驳回申请，则流程结束；
- 如果经理通过申请，则判断请假天数是否超过7天；
- 如果请假天数不超过7天，则流程结束；
- 如果请假天数超过7天，则进入HR经理审批；
- 如果HR经理驳回申请，则流程结束；
- 如果HR经理通过申请，则流程结束。

与之相比，图文并茂的方式明显更容易理解，图1就是将上述请假流程转化为jBPM4流程图的样式。

仅有一张流程图还不够，工作流平台除了提供给业务专家设计流程图的工具之外，还有一项重要任务是将业务流程图转化为可以运行的应用。因此，工作流平台在实际应用中也会分为两个阶段，即流程的设计期与运行期。首先，业务人员和技术人员根据实际的业务流程，使用工作流平台提供的建模语言进行设计，然后得到的业务流程需要发布到流程平台后才能提供给最终用户使用，最终用户会根据已设计发布好的流程模板发起各自的流程实例，从而实现特定业务功能。

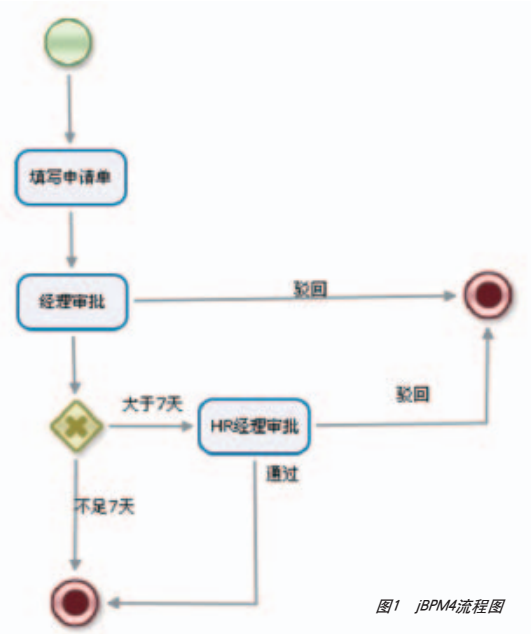


图1 jBPM4流程图

标准图元

在流程设计期，工作流引擎一般都会提供特定的流程设计工具，让用户通过拖拽图元的方式直接绘制出流程图，业务分析人员使用图形化的工具就可以独立进行流程的抽象和设计工作，节省了将业务逻辑再次转述给技术人员的步骤。为了实现这一目标，业界也经过了大量的尝试，其中最成功的成果是对流程中的图元实现规范化，这样做的好处是只要了解典型流程图元的含义，就能很容易画出让其他人理解的通用型业务流程图。

通过对表1三类图元的组合拼装，就可以实现对业务流程的抽象设计，当然在实际使用中还会依据特定工作流平台本身以及面向的特定领域而有所变化，比如在最新的BPMN2规范中就定义了10种不同的事件用以表述更加纷繁复杂的业务流程，而在流程中负责实际功能

表1 三类图元的含义

图元	含义
	圆圈代表事件，如：流程开始事件，流程结束事件，任务超时事件等。
	矩形代表活动，如：需要人工处理的任务，调用某个外部服务的步骤。
	菱形代表此处流程的流向会发生变化，如：根据条件选择分支，出现并发分支，分支流程汇合为一条分支等。

的活动节点，则可能因为使用的语言与环境的不同而支持本地接口调用、动态脚本甚至Web Service或RESTful服务。对于一个完善的工作流平台，考量的将不仅仅是平台本身提供的功能是否丰富，还需要评价是否提供高度的可定制性和扩展性，这样才可以在实际业务出现调整时，信息化系统也能灵活敏捷的紧随其后。

因为XML比较容易来描述有向图，也易于通过dtd或xsd对数据格式进行有效性校验，所以流程设计器通常使用XML来保存设计好的流程图。如上面我们示例的请假流程对应的XML内容就像下面这样：

```
<process name='untitled' xmlns='http://
jbpm.org/4.4/jpdl'>
    <start name='start' g='102,23,48,48'>
        <transition to='填写申请单'/>
    </start>
<end name='end' g='107,433,48,48'/>
        <task g='83,121,90,50' name='填写
申请单' assignee=" #{creator}" >
            <transition to='经理审批'/>
        </task>
        <task g='84,212,90,50' name='经理
审批' assignee=" #{manager}" >
            <transition name=' 通过'
to='decision 1'/>
            <transition name='驳回'
to='end 1'/>
        </task>
    <decision g='108,304,48,48'
name='decision 1'>
        <transition name='大于7天' g='-
20,-10' to='HR经理审批'>
        <condition expr="#{day > 7}" />
        </transition>
        <transition name='不足7天'
to='end'/>
    </decision>
    <task g='217,303,90,50' name='HR经理审
批' assignee=" #{hr}" >
            <transition name='通过'
g='263,457:-4,-100' to='end'/>
            <transition name='驳回'
g='432,325:-77,-32' to='end 1'/>
    </task>
    <end g='409,208,48,48' name='end 1'/>
    </process>
```

其中process标签代表整体流程，内部的每个标签都对应着流程中的一个环节，环节之间的连线则是通过transition标签来标记的，比如对应“HR经理审批”的标签中包含两个transition说明这个环节有两个向外的连线，其中name为“通过”的连线指向end，另一条name为“驳回”的连线指向end 1。

名词解释

与图元类似，工作流平台通常会包含如下几类标准概念：

- 流程定义Process Definition/模板Template；
- 流程实例Process Instance；
- 活动Activity/节点Node/环节；
- 转移Transition/连线Edge；
- 任务Task/工单Work Item；
- 日志log/历史history。

流程定义（或者叫做模板）是流程设计阶段的产物，在业务设计人员对实际业务流程进行抽象设计后，产出的流程定义会被发布到工作流平台中。后续，最终用户根据这些流程定义各自启动的业务流程就叫做“流程实例”。每一个流程定义都会对应着多个流程实例，每个流程实例会指向流程定义中的不同环节，并在运行过程中根据对应环节的预设配置执行特定功能。

jBPM4就将整个工作流引擎划分为下面六大服务接口。

- RepositoryService/**资源库服务**，对应流程设计阶段，用来处理流程定义的发布、挂起、恢复、删除等功能。
- ExecutionService/**流程服务**，对应流程运行阶段，用来处理流程实例的创建、激活、挂起、恢复、终止、删除等功能。
- TaskService/**任务服务**，用来处理人工任务的完成、删除、关联参与者、创建子任务等功能。
- HistoryService/**历史服务**，主要用来查询流程实例运行的痕迹，为流程追踪提供支持。
- ManagementService/**管理服务**，用来管理流程中使用到的异步消息和定时器。
- IdentityService/**人员和组织机构服务**，提供了最简单的组织机构适配功能。

这六类服务接口从战略层面概述了一个工作流平台应该提供的内外部接口，对于实际的开发者来讲，流程引擎内部的推进引擎应该是一个黑盒，任何扩展功能的开发都应该可以直接扩展暴露的接口来实现。

因为流程设计阶段就已经将整体业务划分为不同的阶段，因此工作流平台从设计层面上就在鼓励软件开发人员编写出更加模块化的代码，设计良好的模块可以提供自管理的服务交由流程的对应环节调用。假设在软件使用过程中，甚至是在软件开发过程中，实际业务流程发生了变动，也能进行快速灵活的调整，避免出现牵一发而动全身的灾难性结果。因此模块化也可以算是工作流平台为系统开发带来的额外好处。

通常情况下，工作流程平台都会将一部分常用服务功能进行封装，提供给流程设计人员使用。比如，jBPM4就提供了一套内置的节点来实现最基本的功能。

最常用的Java节点可以用来调用一个标准的Java类：

```
<java name="greet" class="org.jbpm.
examples.java.JohnDoe" method="hello"
var="answer">
    <field name="state">
        <string value="fine"/>
    </field>
    <arg>
       <string value="Hi, how are you?"/>
    </arg>
    <transition to="shake hand" />
</java>
```

如果嫌编写Java类太麻烦，也可以通过Script节点来执行一段动态脚本：

```
<script name="invoke script" expr="Send
packet to #{person.address}" var="text">
    <transition to="wait" />
</script>
```

HQL和SQL节点允许我们直接执行一条HQL或SQL语句，并获得返回结果。

```
<hql name="count activities"
var="activities" unique="true">
    <query>
        select count(*)
               from org.jbpm.pvm.internal.
model.ActivityImpl
    </query>
    <transition to="wait" />
</hql>
```

在jBPM4中我们还可以直接使用mail节点发送各种提醒邮件：

```
<process name="InlineMail"
xmlns="http://jbpm.org/4.4/jpdl">
    <start>
          <transition to="send
birthday reminder note" />
    </start>
    <mail name="send birthday
reminder note">
   <to addresses="johnDoe@some-
company.com" />
   <subject>Reminder: ${person}
celebrates his birthday!</subject>
   <text>Do not forget: ${date} is
the birthday of ${person} </text>
   <attachments>
    <attachment resource="org/
example/birthday_card.png"/>
          <attachment name="picture.
jpg" expression="${picture}" mime-
type="image/jpeg"/>
    </attachments>
    <transition to="end" />
    </mail>
    <state name="end"/>
</process>
```

除此之外，jBPM4还提供了jms、rules、assign节点，可以实现特定业务需求下的不同功能。在实际使用工作流平台时，我们也要根据实际情况，具体选择或开发适合的功能节点去实现对应的功能。

让流程实例进入等待状态

通常情况下，每个流程定义都只包含一个开始事件，当用户决定发起一个新流程时，会调用startProcessInstanceById()方法，根据指定的流程定义id来启动一个新的流程实例。新创建的流程实例会从开始事件节点开始，沿着流程连线顺序向下执行，直到遇到了一个等待活动或到达一个结束事件节点。如果到达的是一个等待活动流程实例将进入等待状态；如果到达结束事件节点流程实例就会终止。

工作流的初衷是为了将可以让计算机自动执行的一系列功能贯穿在一起，自动顺序执行。只有在特殊情况下，计算机无法自行决定流程的流向，这时候需要让流程暂时进入等待状态，直到有外部的操作激活流程实例，流程实例才会继续向下执行。

而在国内的实际项目中，人们接触到更多的却是由大量等待活动构成的审批流程，例如我们在文章前面提到的请假审批流程中，包含的所有活动都是等待节点，我们并没有提供任何一个

可以无需用户干涉就可以自动执行的功能。实际上，像请假审批这类审批流程，基本上不会使用到自动活动。流程实例从创建开始就是步履维艰地一步一停。每当流程实例进入一个活动，就会根据活动的配置自动创建一个任务，这个任务会被添加到对应负责人的待办任务列表中，同时流程实例会进入等待状态。直到任务的负责人调用completeTask ()后，才会激活任务所对应的流程实例继续向下执行。

使用工作流平台所提供的接口API中，最常使用到的两个方法就是startProcessInstanceById ()和completeTask()。startProcessInstanceById ()方法为我们提供了启动一个新流程实例的功能，而completeTask()允许我们通过人工触发的方式告知目前正处于等待状态的流程继续向下运行。

除此之外，工作流平台还提供了state等待活动和与之对应的signalExecutionById()方法，state等待活动与task任务活动类似，都会使流程实例进入等待状态，不同之处是state等待活动不会创建人工任务，因此也就无法采用上面介绍的completeTask()使等待中的流程实例继续执行。取而代之的就是signalExecutionById()方法了，它可以通知对应id的流程实例继续向下执行。很多初学者表示难以了解什么场景下可以使用state和signalExecutionById()。实际上，如果把task人工任务仅仅当做一种普通的外部服务，那么完全可以采用如下方式通过state实现task的功能：

- 为state配置事件监听器；
- 在流程实例进入活动时，触发监听方法，创建对应任务，并将流程实例的id绑定到任务中；
- 流程进入等待状态；
- 当用户完成任务时，使用绑定的流程实例的id来调用signalExecutionById()，通知流程实例继续向下运行。

由上面的步骤可知，task人工任务仅仅是state等待活动的延伸和扩展而已，任务实际上已经是脱离工作流本身的一种特殊的外部服务，只不过因为工作流平台大都需要提供与操作人员交互的接口，因此才特别定义了一套数据接口用于实现人机交互。而state等待活动则不仅可以实现人机交互，也可以在多个系统之间需要相互协作等待时派上用场。

除了上面提到的常用API方法之外，工作流引擎也提供了像endProcessInstance()之类可以直接终结指定流程实例的方法。我们也可以使用suspendProcessInstance()和resumeProcessInstance()来暂时中断、恢复流程实例。注意，与进入state或task活动后进入等待状态的流程实例不同，使用suspendProcessInstance()中断的流程实例不能通过completeTask()或signalExecutionById()之类的方法继续执行。

扩充知识

如果说早期的workflow还是专注于流程推进引擎的算法实现，那么当前的BPM已经是囊括了图形设计、数据持久化存储、异步消息、支持多重服务调用的混合型企业级平台了。实际业务中采用的工作流平台不仅包含文章中所展示的这些基本功能，还可能包含为支撑实际业务流程需求而集成的更多复杂技术，篇幅所限，无法一一详细介绍了。

如果希望更加深入地了解工作流平台的运作，可以考虑阅读下面这些扩展知识。

- Petri Net，流程推进的核心算法。
- FSM，有限状态机是另一类用来实现流程推进的算法。
- XPDL，由WFMC工作流委员会制定的流程规范。
- BPEL，针对Web Service与系统集成方面的流程规范。
- BPMN2，目前风头正劲的流程规范，拥有更丰富的图元语法和表意性。
- XForm，W3C组织制定的电子表单规范。
- HumanTask，基于Web Service实现的人工任务。
- Rule Engine，规则引擎，可用于流程分支判断实现复杂逻辑。
- BI，基于流程数据的统计分析。
- BAM，实时监控仪表盘。



徐会生

翻译过Spring security的官方文档，出版过《深入浅出Ext JS》，参与过jBPM4的开发，目前任职于百度，依然做着JavaEE的开发，希望继续沿着这个方向把事情做深做精。

基于SOA、ESB、BPM实施敏捷的企业IT系统

文 / 辛鹏

企业也好，政府也罢，要做信息化，要做IT，目的是什么？四字即可概括之，“卓越绩效”。这个绩效，可以是外延很大的绩效概念。企业要发展、政府、公共组织也要发展，发展的目标是什么？还是那四个字“卓越绩效”。因此取得卓越绩效是企业、公共组织、政府的共同战略目标。企业的运营发展，都要遵循战略目标。

企业要做什么？要做数据一体化、业务一体化。怎样做？通过技术一体化来做。为什么做？为了实现战略一体化和管理一体化，进而帮助企业取得卓越绩效。



图1 企业信息化的本质分析

如图1所示，从企业信息化的整体来分析，可以分为信息化之What、信息化之How、信息化之Why。

企业信息化之What—做什么？

■ 数据一体化

企业的数据包括MES数据、PCS数据、物流数据、财务数据、ERP数据、CRM数据、核心业务数据等，通过数据一体化彻底解决“数据孤岛”的问题。

■ 业务一体化

企业的业务系统包括CRM、ERP、SCM、财务、OA等，对于制造型企业还有MES、PCS等。这些业务系统之间，需要互联互通，因此必须通过业务一体化来实现这交互的需求，从而解决“业务孤岛”的问题。

企业信息化之How—怎样做？

■ 技术一体化

通过SOA/ESB（SCA、SDO）、BPM/WFM、BI等技术，来实现技术的一体化。SOA作为整体的架构指导原则与思想，在其体系下，通过ESB与BPM、BI等技术的应用实现技术上的互联互通。

企业信息化之Why—为什么要做？

■ 战略一体化

企业战略包括绩效战略、竞争战略、营销战略、发展战略、品牌战略、融资战略、技术开发战略、人才开发战略、资源开发战略、流程战略等，只有将这些不同侧重点的战略，全面管理起来，才能实现整个企业战略的一体化，进而实现企业的创新和发展，使企业成为敏捷性企业。

■ 管理一体化

管理一体化包括核心业务管理及日常办公管理。通过数据一体化将业务系统的产生的业务数据，办公系统产生的办公数据等实现互联互通；通过业务一体化，将企业内部的各业务系统自动关联起来；通过基于数据一体化的统计分析及辅助决策，基于业务一体化的协同办

公，实现管理一体化。

概述

SOA

SOA（Service-Oriented Architecture）面向服务的架构，其中的Service是灵魂，是核心。从Object-Oriented到Service-Oriented不仅是一个名词的改变，更是一种对事物的认知的变化。面向对象强调以“对象”的视角来建模我们这个世界的事物，关键在于对象的识别与创建，主张以细粒度为识别与创建原则，主要应用在系统内的事物自身的建模以及事物与事物之间关系的建模。面向服务，强调以“服务”的视角来建模我们这个世界的事物（具体来说就是企业内的业务逻辑单元），关键在于服务的识别与抽象，主张以粗粒度为识别与抽象原则，主要应用在系统与系统之间的交互事物的建模上。因此，在一定程度上，也可以说，面向对象主内，面向服务主外。当然这里的主内与主外并不是完全割裂开的，在某些情况下，它们两个相辅相成。

面向服务不是对面向对象的代替，更不是面向服务就比面向对象高级了。从软件架构的本质上来讲，它关注的是对组成软件的子系统、组件、模块的划分描述，以及对这些元素之间的交互描述，此处的“交互描述”正是Service的拿手好戏。SOA的目标是构建一个松散耦合的系统架构，可以将企业的应用程序及资源包装为一个个的商业服务，按照特定的合约及规约来请求和使用服务。

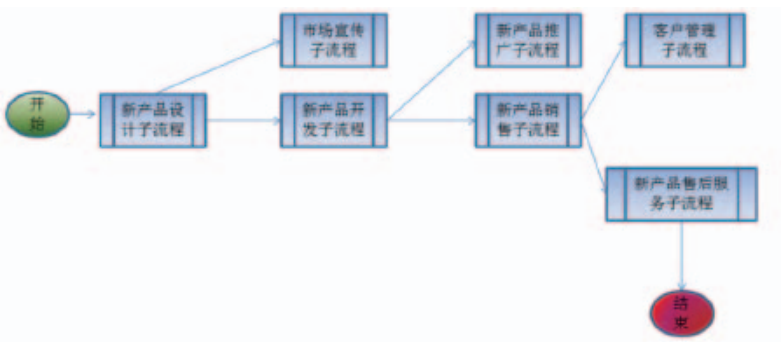


图2 某公司新产品上市业务流程

ESB

企业服务总线（ESB-Enterprise Service Bus）本质上是一个以消息通信为中心的底层基础设施，它负责治理企业的所有应用服务，以连接为导向，为服务请求者与服务调用者搭建起沟通的桥梁。包括服务连接、数据转换、事件转换、服务路由等功能。企业服务总线是保障SOA有效落地的底层基础设施，没有ESB，SOA就只会是一片飘在天上的云彩，最终会烟消云散。因此企业要基于SOA的架构方法论和思想去搭建智慧IT生态群落，就首先要搭建好底层的基础设施。

BPM

业务流程简单来说就是业务的处理流，或者说业务的流转过程。这里重要的不是“流程”这两个字，而是“业务”这两个字。在企业内部，核心业务如下：

- 产品和服务的设计与开发
- 产品和服务的市场营销与销售
- 产品和服务的交付
- 客户服务管理

从上面我们知道了一个企业的核心业务，那么这些业务总要有对应的部门去处理，例如产品研发部负责新产品的设计及研发，市场部负责新产品的推广和宣传，销售部负责新产品的销售，售后部负责产品售后的相关服务，客户部负责客户的管理等，这些部门可以说就是企业的核心部门，是为企业创造价值带来利润的部门。这些部门的工作都是有前后的逻辑关系的，或者说是互相作用的，那么把这些部门的工作按照前后的逻辑关系和互相作用原理，串在一起就形成了一个流，这个流就是企业的核心业务流程，也是企业的价值链，如图2所示。

SOA、ESB、BPM的关系

首先讲SOA与ESB的关系，ESB是保障SOA有效落地的底层基础设施。SOA关注的是企业的整体IT架构，关注企业的各个IT系统怎样交互，它给出这些IT系统交互的策略与规约。而ESB关注的是具体的实现，即提供各个IT系统交互的具体实现，例如一个基于C语言开

发的HR系统通过ESB与一个基于Java语言开发的财务系统进行服务的交互。

SOA与BPM的关系，SOA是从IT架构着手，按照服务的思想来分析及架构企业的IT系统。而BPM是从业务或者说企业的经营管理方面着手，从流程的角度出发，建立并持续改进企业流程。BPM可以及时响应企业的组织架构、业务交互及运营模式的灵活多变，并且可以满足企业内部、企业与伙伴之间的业务交互需求。

由以上可以看出，SOA/ESB的目标与BPM的目标是一致的，即都是通过解决企业的业务交互问题，来提升企业对环境及市场变化的响应能力，进而提高企业的敏捷性。只不过两者切入的角度不同，SOA是从企业的IT架构着手，建立起一个整合业务应用的服务平台，而BPM则是从企业的业务运营着手，解决IT与人、人与人、业务与业务的交互，建立起敏捷的、端到端的流程体系。

实际情况是，没有SOA/ESB之前，BPM产品已经得到了很多成功应用，只不过当遇到跨系统的交互时，采用传统的EAI的方式来解决交互问题，有了SOA/ESB，则可以通过标准的服务来直接实现交互。

基于SOA架构建立的BPM更能展现整合能力及系统的弹性；从另一个角度来看，理想的SOA架构，也需要BPM支撑，SOA中的服务，需要BPM来进行串接。而且通过业务流程梳理，识别与抽象出的业务服务，才是最有价值的服务，单纯的为了解耦而抽象出的技术连接服务，提供的价值是非常有限的。

实施过程及方法

战略阶段

企业战略是是对企业整体性、长期性、基本性问题的计谋。如果企业是一艘船，那企业的战略就是船头，业务流程是船身，战略引导企业到达企业目标，而业务流程则承载着企业的部门、岗位、绩效、业务等，推进企业不断前进。战略是金字塔的塔尖，而战术流程与执行流程则是让金字塔落地的地基。

要搭建企业的软平台，就必须先搭建好

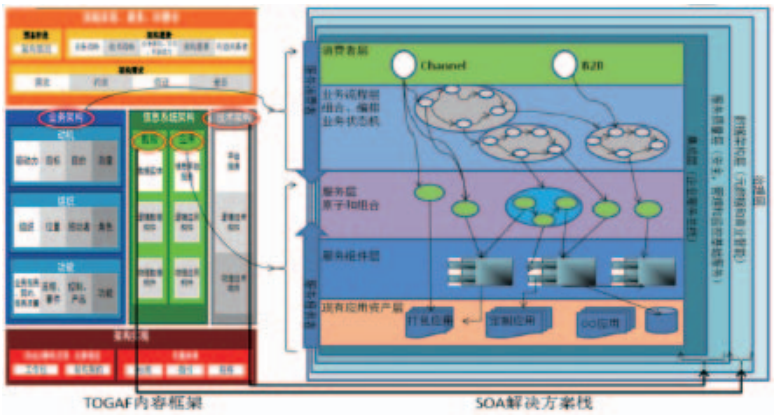


图3 企业架构框架与IT架构（SOA解决方案栈）的映射

自身的企业架构，对于企业架构，不同的组织给出了不同的定义，Open Group给出的定义是——企业架构是关于理解所有构成企业的不同企业元素，以及这些元素怎样相互关联。MetaGroup的定义：企业架构是一个系统过程，它表达了企业的关键业务、信息、应用和技术战略以及它们对业务功能和流程的影响。关于信息技术怎样以及应该如何在企业内实施，企业架构提供一个一致、整体的视角，以使它与业务和市场战略一致。

企业架构从大的层面来分，可以分为业务架构和IT架构。业务架构是把企业的业务战略转化为日常运作的渠道，业务战略决定业务架构，包括业务的运营模式、流程体系、组织结构、地域分布等内容。IT架构指导IT投资和设计决策的IT框架，是建立企业信息系统的综合蓝图，包括数据架构、应用架构和技术架构三部分。企业架构如同战略规划，可以帮助企业执行业务战略规划及IT战略规划。在业务战略方面，可使用TOGAF及其架构开发方法论（ArchitectureDevelopmentMethod/ADM）来定义企业愿景/使命、目标/目的/驱动力、组织架构、职能及角色。在IT战略方面，TOGAF及ADM详细描述了如何定义业务架构、数据架构、应用架构和技术架构，是IT战略规划的最佳实践指引。企业架构是承接企业业务战略与IT战略之间的桥梁与标准接口，是企业信息化规划的核心。

企业架构映射为IT架构

在上节我们知道企业架构包括业务架构及

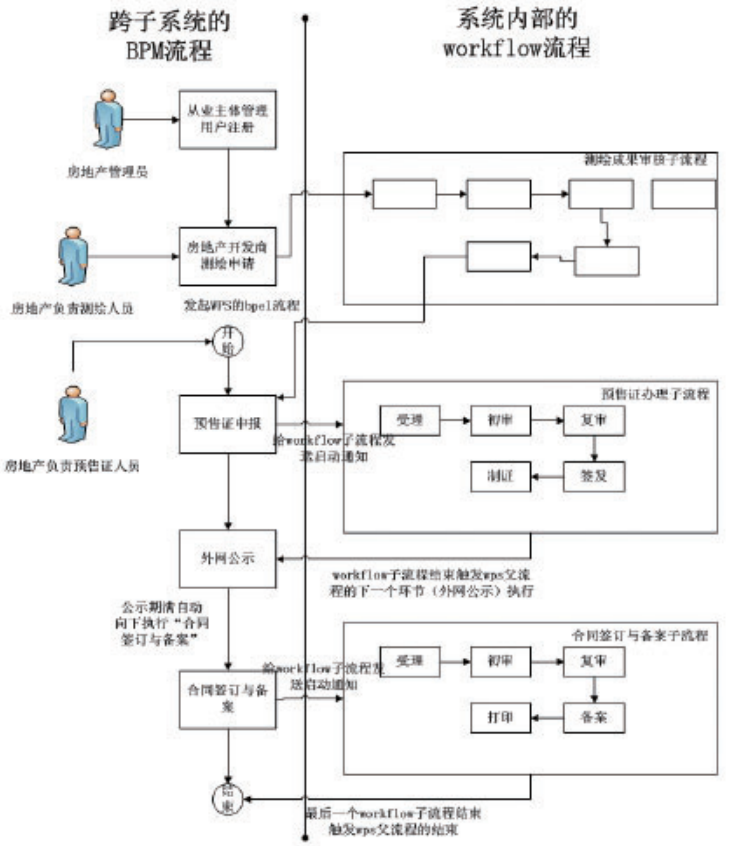


图4 某房管局房地产市场领域之预售相关业务流程梳理

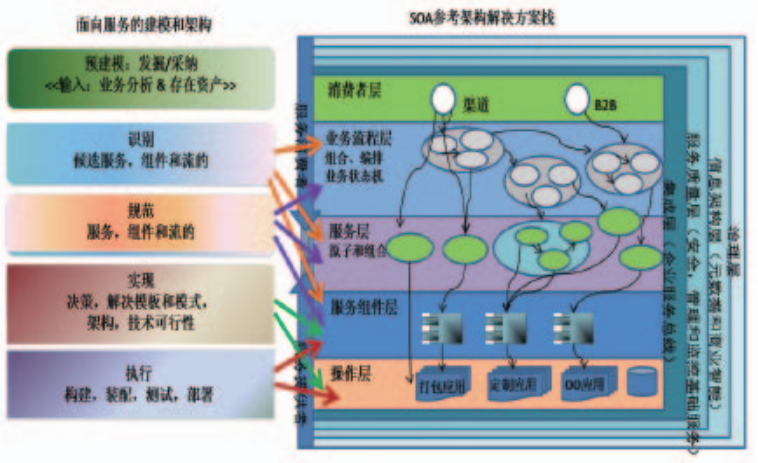


图5 SOMA过程与SOA参考架构解决方案栈的映射

IT架构，IT架构又包括数据架构、应用架构和技术架构。将业务架构转换为IT架构，用IT架构来很好地支撑业务架构是企业信息化最最重要的基础，没有这个基础就无法做好企业信息化，甚至说企业信息化必然会失败。

业务架构及IT架构都有了良好的定义，那么接下来的工作就是将业务架构逐步分解匹配到IT架构，将IT架构进行细化。基于BPM对企业的业务流程进行梳理，在业务流程梳理的基础上去实施SOA，成为业内对实施SOA的关键共识。流程梳理之后，二者形成如图3所示的映射关系。

战术阶段

流程梳理：接下来我以一个某房管局内部房地产市场领域的业务为案例，剖析企业软平台的整个搭建过程。如图4所示。

- BPM负责子系统之间的流程流转，workflow负责子系统内部的流程流转；
- 在此场景中，BPM流程有三个环节：预售证申报、外网公示、合同签订与备案；
- 预售证申报作为BPM流程的第一个节点，先启动实例，同时会启动预售证办理子流程；
- 预售证办理子流程结束时，同时会向BPM流程发消息，触发BPM流程从预售证申报节点——向下执行到外网公示节点，公示节点是一个自动节点（满3天的公示期后），自动触发“合同签订与备案”的环节的实例化，同时向“合同签订与备案子流程”发送启动通知，此子流程结束后，发送结束通知给WPS流程，整个BPM父流程结束。

基于SOMA的服务建模与架构：基于上节的流程梳理结果，采用IBM提出的SOMA方法论，进行服务的建模与架构，关于SOMA方法论，读者可以去查阅IBM的相关文档。在SOMA中，服务的建模与架构分为了如下的4个主要阶段，每个阶段的成果对映射到SOA参考架构的解决方案中。

实战阶段

经过了战略阶段与战术阶段的工作，开始进入实战阶段，在实战阶段，基于SOA、ESB、BPM的企业信息化架构变得更加清晰和具体，如图5所示。

如图6所示，基于SOA、ESB、BPM的企业应用架构的逻辑调用关系，可以分解为如下的调用序列：

- 预售相关业务用户通过B/S浏览器中展现的业务表单使用某个功能模块；

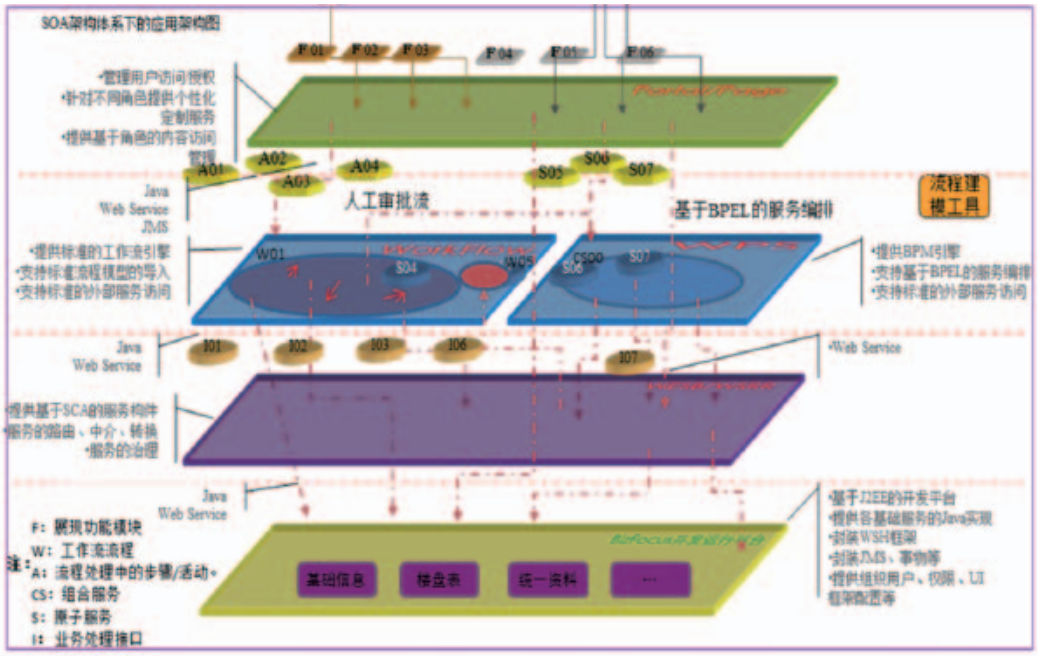


图6 基于SOA、ESB、BPM的企业信息化架构之分解图

■ 预售业务表单中的某个下拉框调用企业服务总线中的数据字典原子服务取得枚举数据；

■ 预售表单通过Action直接调用平台中的业务数据处理接口进行表单数据的保存操作；

■ 数据保存成功后，调用BPM引擎接口，启动人工审批流（例如启动预售许可证审批流程）；

■ 预售证审批流程的某个业务环节直接调用企业服务总线中的某个原子服务（例如楼盘表展示服务、调用行政许可期限计时的服务、查询公示服务）；

■ 业务流程实例的某个业务环节触发Web服务事件（例如触发入库操作）；

■ 业务流程实例的某个业务环节（例如预受理环节）调用BPEL编制成的房屋核查组合服务；

■ BPEL将物业用房核查服务、拆迁安置房核查服务、经适房核查服务三个原子服务编排为组合服务；

■ 审批流程结束后，发通知消息给业务流程的发起者。

总结

本文首先对企业信息化的本质进行了分析，分析了企业信息化之What、How与Why。接着对SOA、ESB及BPM进行了概述，给出了各自的特点以及三者之间的关系。最后按照战略、战术与实战三个阶段，讲述了基于SOA、ESB、BPM来实施敏捷的企业IT系统的过程，以及每个实施阶段的重点理论、方法、工作内容及成果。



辛鹏

东华软件股份公司房管事业部技术副总，北京东方易维软件有限公司CTO，清华大学软件工程专业硕士，中国开放流程社区创始人。
微博地址：http://weibo.com/opug。



Marty Cagan
过去20年，Marty Cagan作为负责定义和开发产品的高级经理人为多家一流企业工作过，包括惠普、网景通信、美国在线、eBay。他亲历了个人电脑、互联网、电子商务的起落沉浮，致力于通过写作、演讲、培训帮助客户打造富有创意的产品。为此，他撰写了《Inspired: How to Create Products Customers Love》一书，创办了硅谷产品集团公司（SVPG）。在此之前，他的最后一份工作是担任eBay产品管理及设计高级副总裁，负责规划全球电子商务网站的产品和服务。

产品探索的意义和流程

文 / Marty Cagan 译 / 林航，王烨，吉绚

Marty Cagan是享有世界声誉的产品管理专家，曾担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作和经验的总结，分享了探索产品的流程和方法。

产品开发宁缺勿滥

有没有见过这样的情形？开发团队刚刚完成手头上的项目闲了下来，可产品经理一想到程序员们自在逍遥，就怎么也坐不住了。为了让他们马上开始新项目，产品经理没日没夜地制作新项目的产品需求文档（PRD），以求速成。这样的事情在产品团队里时常上演，成为产品失败的罪魁祸首。

开发团队就像“嗷嗷待哺”的婴儿，胃口特别大，却分不清哪些该吃，哪些不该吃。只要产品需求文档到位，不论产品好坏，他们就马不停蹄地开始新一轮的产品开发。后果可想而知。

追根溯源，这种现象的产生是因为产品开发团队里的程序员聘金不菲，公司不乐意看到他们闲着，只有“物尽其用”，才会让公司觉得安心。可聪明反被聪明误，这种短视的行为，给产品的失败埋下了祸根。

这种行为还会掩盖开发团队真正的价值。开发团队被公司异化成生产代码的机器，而不是共同探索、打造出成功产品的“战友”。

只让开发人员负责软件的开发，明显低估了他们的价值。正确的做法是先通过产品探索定义出基本产品，确定产品是有价值的、可用的、可行的，然后再交给程序员去开发。

强迫开发团队满负荷工作的公司，显然是管理理念出现了问题：代码好不等于产品好。当然，这种现象和企业文化也脱不开关系。

不可否认，好的项目经理确实能保证产品的交付进度。但不能因此就盲目地让项目管理在公司里唱主角。进度安排合理、资源得到了优化配置不意味着产品就能取得成功。通常，我会建议项目经理在产品探索阶段安静地坐在“观众席”上，等到确信设计的产品值得开发了，再粉墨登场率领团队开发产品。采取这套方案后，收效不错。但如果产品探索阶段，团队里缺少能一统全局的主帅，这套方案就不适用了（在今后的文章中我再讨论这种现象）。

如果开发团队闲了下来，而新一轮产品探索还没完成，该怎么办？其实选择很多。

- 让开发团队利用“余量（headroom）”完善软件架构。
- 让开发团队着手修复产品缺陷，改善产品性能。
- 让开发团队参与产品探索活动。

产品经理给开发团队预留一个月左右的任务。这样即使手头的项目卡了壳（比如原型测试中，用户的反馈意见很糟糕），在想到新方案之前，你的开发团队都有事可做。

开发团队无事可做，还有可能是项目团队里开发人员和产品经理及设计师的比例失衡造成的。如果开发人员过多，那么产品设计团队永远跟不上进度，开发团队必然会“档期”不满。

要记住，产品经理的使命是确保团队开发有价值的产品。让开发团队盲目开发未经验证的产品，也是一种浪费。

产品探索计划

为了不让程序员闲着，产品经理把本该花在产品探索上的时间都花在了赶写产品

需求文档上。可这样的产品探索毫无价值。时间再紧迫，也不能牺牲那些创造价值的关键步骤。而有些公司非常重视产品探索，却没有对这个过程做出合理的规划，不能保证每天做的都是有用功，自然无法快速开发出基本产品。所以，宝贵的时间就这么白白浪费了。

产品探索规划是否详细，取决于公司的文化氛围和产品探索人员的能力。很多公司（特别是大规模公司）中，团队需要一个产品探索计划，明确项目必须实现的内容、需要的资源，并给出粗略的进度安排。

我将列出常见产品探索计划的组成部分。但这不是模板，不要照着里面的内容生搬硬套，做些不需要做的事情。我列举的每个部分分别解决不同的问题，你可以根据你要解决的问题，选择合适的探索计划。作为产品经理，你要审查这个计划并确保团队按计划执行。

■ 核心探索团队：确认项目的产品经理、首席设计师和主程序员，并且确保这些人员在探索产品的过程中随叫随到。

■ 探索扩展团队：谁将给这个核心团队提供支持？你是否需要视觉设计师？你需要原型设计师的协助吗？需要可用性测试资源吗？需要有特定开发经验的开发人员吗？需要QA团队提供人手吗？需要市场部的人员支持吗？

■ 关键参与人员：这个项目必须得到谁的批准（即谁拥有“否决权”）？此外，还可以写上你认为能够对这个项目发表独到见解的聪明人名单。

■ 客户发展计划：这个项目需不需要特约用户？如果需要的话，谁是牵头人？如果不需要的话，你会请哪些客户（用户）验证你的产品理念？

■ 主要风险：这个项目的主要风险有哪些？你将如何解决这些问题？

■ 用户研究工具：这个项目需不需要创建一些人物角色？需要做网站检查和商业分析吗？

■ 用户测试计划：你打算如何让实际用户测试产品？初步测试计划是什么？

■ 产品战略（愿景）：是否需要在制定项目细节之前，考虑项目所在领域的长期发展？

■ 产品原则：产品的所在领域是否需要独创一些产品原则？

■ 支持文档的程度：核心团队需要在以下方面达成一致意见：在这个项目中，需要哪些额外文档、需要做到什么程度，包括用户故事、用例、测试计划等。

制订计划只是产品开发中的一步，更重要的是保证产品探索确实是朝着满足基本要求的产品方向努力。另外，将这两步结合起来后，什么时候能真正开始构建产品也是关键。这个过程的监督工作包括两个部分。

首先，我认为，产品的负责人（如产品经理或首席用户体验师）需要对这项活动尽到监督责任，他们积极的监督会对定义产品起到积极的作用。这里要说明一点，这里的“监督”指的是每周至少询问一次产品探索团队下面这些问题。

■ 你这周与什么样的客户和用户进行过比较有效的沟通？

■ 你从你们的讨论中了解到了什么？在你了解它们的过程中，有什么事情使你感到惊讶吗？你有了什么新的见解？

■ 你是否遇到了潜在关键点？【注：参见http://www.svpg.com/your-business-plan-is-wrong】

■ 根据你所了解到的内容，下一次你会进行哪些新的尝试？

■ 开发人员最后一次评估产品设计和客户反馈信息是什么时候？

■ 开发人员在项目可行性和可能的解决方案上有什么意见和反馈？

■ 用户认可这种想法的价值吗？如果不，那么为什么他们不认可？

■ 目前原型最主要的可用性问题有哪些？你打算如何纠正这些问题？

■ 从可行性的角度来讲，开发人员最关心原型的哪些方面？针对这些问题，你有哪些应变措施？

■ 最后，你认为在两周之内完成一个满足最低要求的产品的可能性有多大？如果有困难，那么团队是应该继续努力还是考虑开发其他产品？

其次，产品经理也有监督其他事务的职责，比如在项目管理方面，需要确保产品探索

计划的实施情况，做好开发的一切准备工作。具体来讲，这包括保证开发团队有足够的资源开展工作，明白项目时间限制及做好时间管理，确保开发者、产品管理者和设计者之间的良好沟通，根据计划从整体上推动进度，以及确保项目中的时间都被有效利用。

项目经理的确可以督促大家完成项目，但要确保先确定基本产品，而不能仅写些开发文档。

不管怎样实施计划，如果公司很开明，允许你在探索产品上花时间，而不是仅仅撰写需求文档，那么我希望你不要浪费产品探索中的每一天。如果你已经花了很多工夫，但仍感觉到很难开发出好产品，你应该尝试制订产品探索计划。

产品探索日记

产品经理和设计师们放弃原有的线性、瀑布式的开发流程，转而采用我倡导的这种更具迭代性、探索性的流程后，通常需要花一些时间才能适应它的快节奏，掌握产品探索的韵律。

这篇文章将再现产品探索的情景，让大家了解其核心内容。

因为产品的开发过程涉及的因素太多（如产品类型、投入的精力等），做这样的概括并不容易。我尽量使用比较典型和普遍的情景。为了覆盖尽可能多的要点，我构思了以下情景。

第一周

周一

■ 为了明确商业目标，我和项目的主要负责人一起进行了产品的机会评估与讨论。

■ 与项目的首席设计师及主程序员开第一次讨论会，制订出产品原则。

■ 紧接着，我们进行了头脑风暴，想到不少好点子。

■ 与首席设计师一起创建第一轮关键人物角色。

周二

■ 和首席设计师继续优化前一天完成的用户角色，确定主要和次要角色；讨论用户使用产品的情景，并列出主要情景和次要情景。

■ 与用户研究员一起明确潜在特约客户名单；通过电话联系特约客户，筛选候选人。

■ 首席设计师根据讨论结果快速创建产品原型。

周三

■ 我、首席设计师以及主程序员一起检查前一天完成的产品原型，看它是否符合我们最初模拟的几个用户情景。讨论非常激烈，原型远比我们设想的要复杂。

■ 我们回顾之前提出的产品原则，再一次明确了重点，之后大幅简化了产品原型。

周四

■ 我、设计师和项目的主要负责人一起查看产品原型。这个原型还在初期，但是已经包含了三个最重要的使用场景。通过讨论，我们意识到我们和项目负责人的想法有些脱节，因为双方对用户的理解不同。

■ 继续电话联系目标客户，确认6家公司成为我们的特约客户。最棒的是，他们似乎对我们的产品将解决的问题有极大的兴趣，因为他们的公司都还没有找到好的解决办法。如果我们的产品能够解决这个问题，那么他们会非常看好我们的产品。

周五

■ 我们和主程序员一起评估修改后的原型。他对主要功能的可行性持谨慎乐观的态度，并提出了两个重要的问题。第一，其中一项功能的开发成本可能颇高，而且执行速度很慢。他需要点时间研究这个问题。第二，他认为我们可以省去一些收集数据的工夫，因为我们需要的数据可以从数据库提取。

■ 咨询有关法律人士，向他展示原型，确保模型没有和法律相冲突的地方。他们认为其中一处需要进行详细评测，检查结果会在下周给出。

第二周

周一

■ 早上，我、首席设计师以及项目负责人一起拜访第一个客户，请四位潜在用户试用产品原型。这次测试收获颇丰。可惜原型的表现与我们的期望还有很大的差距。我们发现自己此前对用户的理解还很肤浅，项目负责人的理解也不完全正确。现在我们对用户的理解又进

一了步，而且达成了基本的共识。

■ 回到公司后，我们立刻讨论如何修改原型。我们简化了自以为重要的场景，修改了术语，去掉了那些根本不会被用到的高级功能。

周二

■ 主程序员在研究完高风险的技术问题后，确定他之前的担心是有道理的。他提出了新的可行方案，可以实现同样的效果，而且开发难度更低。于是我们根据他的意见对原型进行了调整。另外，他确认了我们本来要收集的数据在数据库中都能找到，这就进一步简化了原型。

■ 下午我们拜访了另一位客户，请三位用户参与了测试。今天的情况比昨天要好很多。我们仍然不确定原型的交互设计是不是足够好，但至少用户能够完成主要任务了。之前去掉高级功能的这一步棋确实走对了，因为用户并不关心它们。我很高兴我们坚持了以客户为中心的价值观。有两位测试用户希望我们第一时间通知他们产品完成的消息。

周三

■ 与首席设计师、主程序员碰头，讨论目前的成果和接下来的工作。主程序员又提出了一项针对关键流程的优化建议，恰好解决了首席设计师面临的几个棘手的问题。

■ 我和首席设计师向视觉设计师提出了我们对视觉设计的几点想法。经过先前的用户测试，我们认为必须将产品中的两个基本概念用某种方式清晰地传达给用户，并且给出了关键状态和用户可能做出的动作。视觉设计师说她已经有了可以表达这些概念、提升产品价值的想法。她会在接下来的一两天内给我们一些具体的方案，之后我们可以为这些方案提建议。

周四

■ 我们仍然在努力定义基本产品。我们相信已经把最关键的功能都涵盖在内了，但不确定到底要不要再加上一个比较酷的功能，不知道它是不是必需的。我们计划暂时从模型中删除这个功能，看看测试的效果如何。

■ 下午，我们拜访了另一位客户，请三位用户测试了产品原型。原型的可用性已经很棒了，但删除了之前说的那个功能后，原型没有得到用户的热烈响应。我们又展示了之前的版本，这次他们表现出了极大热情。我们得出的结论是，这个功能确实非常重要。我们的原型中已经尽可能地删掉了所有不需要的部分，但仍然能激发人们的购买欲望。我们去除了很多高级的功能，简化了关键流程，应该能在主程序员估计的时间内完成开发任务。

周五

■ 我、首席设计师与可视化设计师一起讨论了她设计的方案，我们特别喜欢其中的一个方案。稍做调整后，我们把它用在了原型上。

■ 主程序员根据现有（也可能是最终）的原型估计出了产品开发的周期。这个时间可以接受，并且有风险的技术问题已经排除。

第三周

周一

■ 我们将包含最新的产品原型展示给项目负责人，并且总结了最终的产品设计（包括最新的界面设计），以及所有来自测试客户的反馈。她很喜欢这个产品，希望旁观下午的测试。她也想就价格和定位询问客户的意见。

■ 下午，我、首席设计师以及项目负责人带着最终原型去见另一位客户。我们请4~5位用户进行了测试，得到了理想的反馈——没有遇到使用问题，对功能和价值的反响也很好。所有用户都很期待这款产品的问世，也愿意推荐给朋友或者同事使用。

周二

■ 继续完善原型，并将这些天来了解到的关键点录入项目维基。

■ 向给更多的项目参与人展示了产品原型，包括法律顾问、市场部、产品管理和工程方面的副总。他们提出了一些问题，但都认可用户测试的反馈。

周三

■ 我、首席设计师以及主程序员碰头，讨论如何撰写产品文档，方便开发人员、测试人员、部署人员开展工作。

■ 我和首席设计师分工撰写文档，我们将各自负责的部分写到项目维基里，并添加了一些对产品原型的链接和注释。

看完以上日记，希望你能明确以下10点。

■ 在项目开始时，确定你对目标有清晰的理解（产品机会评估）。

■ 从产品探索最开始时就建立和首席设计师、主程序员的密切合作。

■ 注重真正的协作交流，而不仅仅满足于写那些没人看的文档。

■ 迅速将关键的产品理念融入到原型的制作中。

■ 设法验证自己对产品的假设，不要纸上谈兵，尽快判断哪些假设正确，哪些不正确。

■ 尽早反复地将产品原型提供给目标客户测试，及早发现问题。

■ 记住要确定基本产品。

■ 产品探索的目标：确定产品是有价值的、可用的、可行的。

■ 在产品探索的过程中不断保持和相关部门的信息互通，向大家展示不断更新的产品原型，让他们了解进度。

■ 不要把时间花在为开发人员写文档上，等到产品确定后再写。

当然了，实际项目不一定完全与我的描述相同，我希望传达这样一种观念：在把产品理念展示给真正的用户之前，不应该花两周时间撰写产品文档，更不能花三周时间制作初期产品原型。

产品探索有特定的节奏和规律，它以构思产品设计、原型制作、用户测试，以及最重要的一点，反复学习为基础。如果你还没有亲身体验过探索产品的过程，那么我希望这篇文章可以让你大致了解产品探索是什么，以及它与直接开始撰写产品文档的做法的区别。



本文节选自华中科技大学出版社《启示录：打造用户喜爱的产品》一书和作者的博客。该书从人员、流程、产品三个角度介绍了现代软件（互联网）产品管理的实践经验和理念。特此感谢华中科技大学出版社与Marty Cagan先生授权。

业界报道

从专业到非专业，也是一种“突破”

2011欧特克AU中国“大师汇”侧记

11月16日，欧特克主办的创新设计领域年度盛会——AU中国“大师汇”在京举行。本届大会以“突破的力量”为主题，就工程建设、制造业、传媒娱乐等各行业涌现出的创新理念与实践真知进行交流、探讨，涉及云计算、移动、可持续设计等。

相信很多人的耳朵已经被这些话题磨出茧子来了，而会上的一个不起眼的话题，却引起了参会者的普遍兴趣——Autodesk推出了面向大众消费领域的一系列产品，比如iPad雕刻程序123D Sculpt、在线绘图网站SketchBook速写簿、在线室内设计家居装修软件美家达人等。那么，欧特克为何想到要做非专业产品呢？在从专业向非专业转移的过程中，有没有出现水土不服？是怎样克服的？

欧特克中国研究院核心产品和运营资深总监云浪生对此做出了解释：“我们之所以做面向大众的产品，实际上是时代的产物。首先，原来做设计需要一定的计算机能量，而当年在阿波罗登月计划里所有的计算机能量加起来还不如现在一个iPhone，一部iPhone的能量就足以支撑起原来需要很大能量才能支撑的产品。这便为我们做非专业产品成为可能，我们可以将已有的技术放到更适合于大众用户使用的平台上。其次，现在技术平台越来越强大，专业和非专业的界限变得越来越模糊。由于技术的发展，原来专业的东西，现在一般人也可以做，而且还有传播渠道。”

当然，这个转变的过程中也遇到了一些挑战。“专业用户人数规模稍小，而非专业用户人数极为庞大，怎样做这个市场是一门学问，我们还在这方面进行恶补”，云浪生说：“但从另一方面来说，我们也有一些优势，毕竟在设计领域做了那么多年，形成了一些技术沉淀，而且同种类型的产品，我们只做一款，这样就比较容易形成合力。”

最后，云浪生分享了他对非专业产品的设计理念：“在确定做一个产品时，首先要考察它可能会有多少人使用、有什么价值，如果只有少数人在用，那就没做的必要了。因此，覆盖面多大是最重要的。接下来才是怎样把它做得简单易用，让更多的人用。在具体的实现时，使用什么技术和平台，要综合考虑应用场景的情况。”


云浪生认为做非专业产品是时代的产物

产品经理要学会说“NO”

文 / 张勇

当遇到不合理或者不切实际的“想法”时，产品经理应该首先考虑这个想法到底能否实现，这才是产品经理的思维模式。每个产品经理都应该满怀自信地向每一个“想法”大胆地说一声“NO”！

前段时间一位从事产品经理工作的朋友向我诉苦：业绩压力大，产品问题多，渠道推广不利，盈利能力不高，团队不好管理……之后问我：“产品经理的真正价值是什么？”这问题让我有些意外，但好像又在意料之中。意外的是看到了中国的产品经理在成长，能在实践中遇到问题后积极思考，并试图解决问题；意料之中的是，这又是目前存在于产品经理中的共性疑问。

相信很多产品经理都和这位朋友一样，能说出工作中的一大堆问题，也有很多产品经理说企业没有给自己机会，不能展现自己的能力，所以工作不好做甚至由于问题太多产生了对产品经理职位前途的质疑。

现实真的有这么严重吗？还是我们忽略了什么才引发这一系列问题？这是我们需要思考的，也是我们在产品经理职业发展之路上必须面对和解决的关键环节。

问题到底出在哪儿？其实这位朋友已经给出了答案，就是很多产品经理并不知道作为一个公司的产品经理真正的价值是什么。

让我们先来分步骤剖析产品经理的核心价值到底是什么。

典型的产品决策流程

我们模拟一下大部分企业具有典型性的关于产品的决策流程。

老大提出构想→产品经理把构想规划成产品→研发生产实现产品本身（介质）→市场、销售实现产品交换。

这个流程我们再熟悉不过，产品经理在其中扮演了一个角色，就是实现老大想法的一个执行人。这个内部的业务流程有问题吗？对于流程本身来说，可能问题不大。大部分企业都是按照这种线性的业务流程来操作的，这证明它还是比较简单实用或者说习惯采用的一种业务流程。

但为什么在这个流程下，会遇到很多问题？比如当产品投放到市场后发现与之前的预期大相径庭，又比如研发部门说这个产品不好实现或者不切实际，再比如销售说这样的产品根本卖不出去。也许最后老大还会问，产品经理是怎么搞的，这么好的一个产品怎么搞成这样。如此如此，产品经理便陷入补漏、救火等这些让其焦头烂额的无尽痛苦深渊中，无法自拔。相信有很多产品经理现在处于这样的状态。肯定有人会问，这是为什么呢，问题出在哪里？

问题首先出在产品经理自己身上，作为产品经理，我们要对产品经理的性质工作有准确的认识，产品经理真的是单纯的执行业务角色吗？我们每天都说产品经理是“军师”、“产品的CEO”等，但我们是否由此真正的理解了产品经理的工作性质？当然从这么多的问题看出，其实我们并没有准确的认识。

产品经理的双重职能

那产品经理在企业中到底是一个什么性质的角色呢，我先给出答案**“产品经理是具备战略制订和战略执行的双重职能的管理职位”**（此处的战略指产品战略，而非企业战略）。这里我先做个定义，然后再说明原因。

为什么产品经理是具备战略制订和执行的双重职能的角色呢？我们做产品目标是什么？赚钱？不全面。满足用户需求？太理论。那到底是什么呢？

首先我们的企业一定有企业战略（忽略没

有企业战略的企业），企业战略是支持企业愿景（发展）的路线规划。企业愿景由企业战略支撑，那企业战略靠什么实现呢？其实就是产品，通过创造一个符合企业战略的产品在市场上进行价值交换来实现企业战略。

说到这里也应该明白了我们做产品的目的，就是用来实现企业战略的。

做企业和做产品的思路是一样的，企业发展需要战略，那我们的产品需要战略吗？答案是肯定的。

用一张图表来说明产品经理的工作内容，如图1所示。



图1　产品经理的工作内容

我们可以看到，产品经理的工作主要由三大部分组成：战略活动、规划活动、战术活动。这其中又包含十二项工作内容。从工作内容上我们就可以看到产品经理是一个涉及战略制订和执行的角色，所以说产品经理的职能具有双重性。

产品经理可以比较一下，现在的工作内容是否全面或者缺失了哪部分工作。也许缺失的这部分正是问题的关键。曾经遇到这样一家企业，他们做一个高校服务类产品，老总在这方面有多年个人经验，所以提出要做这样一个产品，然后产品经理策划了很多的推广手段，做一些会议、搞一些活动、拍一些视频等。其实这个产品经理一定不清楚自己的工作内容是什么，在这种情况下，产品经理会从某个方面，或者是如图1所示的商品化、技术化或者市场化入手来开展工作。这样问题就来了，老总提出的可以称作一个“产品”吗？在我看来，那也许只能算作一个“想法”或者“点子”。离真正的“产品”还有很大距离。如果产品经理直接拿着一个“想法”去当产品做，可想而知接下来的问题会有多少。

因为在工作内容缺失的情况下，产品经理没有明确解决涉及产品的一些关键问题：我们的产品到底是什么、我们的价值是什么、我们的产品竞争优势在哪里。如果不解决这些关键性问题，那么这个产品只有失败的命运。这就是工作内容缺失带来的问题。作为产品经理首先要明白自己的工作内容是什么，这是基础。

产品经理更重要的是一个思考者

当然，知道了工作内容还不足够，我们一定要知道每项工作的价值和意义。换句话说就是产品经理的工作也是按照流程“步步为赢”的，如果在关键环节或者工作开始时就出了错，那么，接下来就是一错再错，这是需要每个产品经理着重关注的一点，也是保证我们产品能够成功的前提

那产品经理的每部分工作到底有什么意义和价值呢？如图2所示。

这里可以看得出，产品经理的工作无非就是三个内容：知道做什么、知道怎么做和去做。到这里我们应该可以发现一些我们产品经理可以体现自己价值的地方，那就是“知道做什么”。这是一个非常重要的工作。他决定了我们产品的方向，决定了产品的成败。

然而很多产品经理是老大怎么说，我就怎么



图2　产品经理工作的意义

做，老大指到哪儿，我就打到哪儿。这样的话，我们作为“军师”的角色就显得无关紧要，何谈价值。而且我们作为要为这个产品负责的人，你是否确定老大说的就一定是对的（往往老大提出的只是一个想法或者创意点，我们却把它当做产品来做）？如果不对或者有更好的方案呢？这些会直接影响到产品经理的后续工作，甚至是诱发产品生死的关键。所以产品经理在认识到自己工作内容的基础上，更要明白让我们产品成功的最关键环节在哪里。第一步就是明确产品的目的和方向，是否符合企业战略，是否符合企业现有资源条件，是否真的有市场，是否可能比竞争对手做得更好，是否真的可以成长为一个可以持续盈利的产品（拿项目当产品做的另论）。

那产品经理的价值是什么呢？在我们有了全面的认识和意识以后，我们就会知道，作为一个产品经理的最大价值就是可以给老大（企业）一个更符合现实、更符合企业利益的完整解决方案。对，就是解决方案！也是产品经理“战略活动”工作中的核心价值的意义所在。

企业或者老大不需要找问题，问题人人都知道，他需要一个可以解决问题的合理方案，这才是产品经理最大的价值。公司内没有任何一个职位或职能部门能够站在全局的角度和产品本身的角度去考量这个问题，只有产品经理。

产品经理是建议者、执行者，但更是一个思考者，能够拿出方案建议的思考者。

因此，每当老大（公司）提出一个想法时，便是产品经理体现自身价值的一次机会。回头想想我们是不是错过了很多次这样的机会，错过了能够体现一个“思考、建议”者价值的机会。如果有人再说公司没给机会，不管你们信不信，反正我是不信。

有“NO”就大声喊出来

既然知道了我们的机会在哪里，按照产品经理的思维模式，我们就能很轻松地面对每次老大提出的想法，因为想法就是想法，落地成为产品还需要一个很长的过程，我们在其中有很大的空间可以发挥我们的职业优势去实现我们的价值。

有一个数据说，中国的产品成功率只有4%左右，这其中最大的问题就出在我们把一个想法当作了产品去做，导致后续问题堆积无法协调，最终导致产品死亡。也就是说，大部分老大提出的想法是不成熟或者不全面的，需要产品经理去分析、研究、验证这些想法，然后现实地形成相关的产品；或者这些想法根本就是没有前途的，我们却把这种没有前途的想法落地成了所谓的产品，也就是这种做法造就了4%的低产品成功率。4%也说明我们的产品经理确实没有尽到应有的职责。这是需要我们认真对待和反思的。

当然企业也有很多问题：短期利益驱使、跟风逐浪心态、管理职能过多渗透等。这些是影响我们产品经理工作的一些因素，但回头一想，我们在面对这些问题时的态度是怎样呢？我们又去做了些什么呢？我们是否能够发现其中的问题，是否尝试过用更科学合理、更符合市场规律的方案去和这种不合理对抗呢？我想企业无非是一个逐利机构，企业利益是每个人能够理解和意识到的，如果产品经理能够拿出基于企业利益，基于企业发展的更全面的解决方案或者建议，我想再强势的老大（企业）会从此改变对你的看法，也会认同你的价值。

回到饭桌上，当我听到“产品经理的最大价值是什么”这个问题后，略作思考便回答“产品经理要学会说‘NO’” 这便是产品经理的最大价值！

当遇到一个不合理或者不切实际的“想法”时，我们应该首先考虑的不是如何实现这个想法，而是这个想法到底能否实现，这才是产品经理的思维模式，一个职业的产品经理人应有的态度，人云亦云非我产品经理所为也。作为产品经理，我们应该运用职业特点和专业能力为企业提供最好的解决方案，这才是关键！

希望从今天开始，每个产品经理都能够满怀自信地向每一个“想法”大胆地说一声“NO”！



张勇

具有多年从事互联网社区和搜索领域的产品管理工作的经验。

责任编辑：董世晓（dongsx@csdn.net）



张庆化
北森CTO

放任型的教练

北森CTO张庆化专访

记者 / 董世晓

在担任北森CTO之前，张庆化曾任职于Yahoo!和MySpace，拥有深厚的技术积累，而在管理方面，他把自己定义为一名教练。

《程序员》：据我所知，您曾在Yahoo!和MySpace工作过，当时是怎样的情况？

张庆化：Yahoo!是我的第一份正式工作，从初级工程师，直到最后担任了搜索竞价事业部架构师。那时主要负责竞价排名后台财务系统的计费管理，因此整个系统的设计在事务完整性和一致性方面的要求特别高。在技术方面主要是基于SOA架构体系，采用COM+进行事务管理，以保证事务完整性；在业务方面，通过实时数据和备份数据的同步，解决线上系统和内部系统的数据操作，采用MSMQ（微软消息队列）进行消息同步。

后来，我加入了MySpace中国创业团队，开始时主要去美国洛杉矶MySpace总部将网站迁移到中国，后续还负责分布式缓存DataRelay以及开放平台的搭建。MySpace本身相当开放，它所有的代码权限对工程师都是开放的。我也是从MySpace开始，才真正接触到完全互联网级的开发。在我看来，MySpace是在.NET技术领域使用微软解决方案最成功的一家企业。与在Yahoo! 所从事的偏重于企业软件不同， MySpace更多的是性能、海量数据、并发等方面的要求。最火爆时，MySpace同时支撑在线用户数达2亿多。

《程序员》：为何后来选定加入北森？

张庆化：读大学时，我就曾在北森做过兼职。当时北森刚起步，员工不到10人。在那段时间，我学到了很多，也与创始团队建立了比较深的感情。在后来的日子里，每年我都会和北森的老板交流一次，了解双方的进展，说句不太恰当的话，我是看着北森一步步发展起来的。在2009年的那次交流中，北森老板提到说想要做人才管理SaaS平台，而当时SaaS还是新鲜事物，出于技术人员喜欢挑战的天性，我对这个项目产生了浓厚的兴趣，于是顺其自然我就加入了北森。

《程序员》：那确实是很早就开始云计算实践了，这个过程怎样？

张庆化：大家都说在做SaaS、云计算，但具体到真正的行业业务领域，国内并没有几家有成功的业务模型，更多的是在做存储或虚拟化。我们的人才管理云计算平台，让用户真正通过互联网就能使用我们的服务，这就意味着必须要满足高性能、高并发等互联网技术的底层要求。而我们是面向企业的，因此必须按照企业软件的特点去实施，例如灵活定制、多租户隔离等。通过不断摸索，我们最终形成了是 “人才管理云计算平台架构=互联网底层架构+企业软件+运维监控平台+云安全”这套架构。

《程序员》：在研发这套架构的过程中，您都遇到了哪些挑战？怎样解决的？

张庆化：挑战表现在技术和人两个方面。必须承认，有挑战才有可能产生价值。

技术方面。首先，技术框架设计挑战。设计开发人才管理平台之初，更多的是SaaS，云计算还不流行，但应用场景已经很清晰了。将我们的三千家客户迁移到平台上，意味着系统会有上万人同时访问，所以在搭建体系框架之初就得考虑高性能和高并发双重因素。

其次，企业软件一个重要特点就是个性化，这要求我们要确保租户隔离，在保证灵活定制的基础上确保企业信息的独立性和安全性。最大的挑战也是最关键的设计就是如

何把两者集成到一起。当然，原则是既定的：互联网的技术用于体系底层搭建，在此之上考虑企业软件的灵活性。最后形成了互联网底层架构和企业多租赁架构深度整合。

再者，时间挑战。公司的目标是在一年内推出平台。这就意味着所有产品线都要同时铺开才能确保如期完成。为此，我们建立了一套开发模式保证体系：利用互联网的开发模式缩短开发周期；确定好底层架构，在此基础上支撑六条产品线同时进展；产品方面以迭代模式进行，先做出一个模型，然后在这个模型的基础上进行不断的调整。还有，技术体系挑战。我们要做到整个底层架构体系要相对独立，不能够和我们的业务体系牵扯得很深。就意味着我们在做底层划分时，要体系支撑产品线，但不能制约产品线。

最后，运维挑战。以前当线上系统出现问题时，都是客户打电话过来我们才知道。现有专业的SystemOPS团队和AppOPS团队，能够高效地保障系统7×24小时正常运转。通过我们开发的“鹰眼系统”来监控线上业务系统，在线上出现问题时，第一时间收集到线上环境状态并解决。

人的方面。首先，最大的挑战是工程师的思想意识。例如互联网产品会大量使用开源技术，但开源技术可以用在什么场景，需要工程师有这个意识。为此我们采用了敏捷项目管理模式，当遇到问题时工程师会贴到白板上，定期进行评估和指导。还有，我们在进行产品设计时遵循了CAP原则，这让我们的工程师形成了一个认知体系——开发的服务必须可降级，所以每位工程师在设计体系时，需要评估服务能否能做到降级、当这个服务降级时是否会影响到其他服务。这些都在思想意识层面挑战着工程师。

其次，促进工程师成长。在海量数据时代，很多业务场景都需要大量的计算，传统的方式在实时处理方面已捉襟见肘。为此，我们将有经验的工程师召集在一起，进行头脑风暴，拓展思路讨论出可行的方案，并确定负责人去实现，大家可以主动认领。而当这个解决方案得以实现时，会给予工程师奖励，比如微软TechEd门票。用有挑战性的工作促进工程师的成长。

在工作中，如果工程师有任何疑问，可以随时来找到我讨论。有的时候，我也会直接认领任务，编写代码实现。

《程序员》：就是说您还在做编码工作？

张庆化：是这样的，这可能与国内其他公司的CTO不太一样。我对整个技术团队的定义是：技术团队的任何一个人，当然包括我在内，都没有纯粹管理者的工作，都必须能够以身作则地从事真正的开发工作。我现在还会把30%的时间花在写代码之上。我始终认为，只有切身参与到团队的实际工作中，才能做好对公司整体的技术把控力。

《程序员》：既然您不把自己定义为纯粹的管理者，那您是如何开展管理工作的？

张庆化：在团队成立之初，更多的是讲解我的经验，亲自编码把体系架构写出来，给成员做出一种模型来学习、沿用和发展。随着我们体系的逐步发展，我的管理风格也发生了变化，目前是“放任型”的，“放任”并不是放任不管，而是建立在原则加信任两个基础之上的。

互联网企业的特点要求我们必须“快”，更多地建立原则，尽量减少流程类等约束发展的东西。我们有各种原则，比如 “线上稳定大于一切”。因为我们是提供云计算服务的，当线上产品出现问题时，工程师必须停下手头工作去解决线上问题，以保证给客户提供更快速、更优质的服务。还有其他原则，比如“言出必行”，这就意味着执行力、项目Deadline不许变。如果是由于需求没有规划清楚，我们允许可变的是过程中的需求，不可变的是完成时间，这样才能确保我们的进度。因此，我在不断完善技术体系，制订好开发规范，在过程中尽量地减少流程。

“用人不疑，疑人不用”，我信任团队的每一名成员。这样使大家工作起来有更多的自主性，工作效率会更高。我们有明确的白板文化，每天的站立会议，我也会定期去参加，当然我只是旁观者。我现在更多的是他们的一个“资源”，他们有什么疑问随时可以找我来协助解决，包括编码。我的期望是我们每一个团队都能自主性地驱动各种事情的发生，朝着自组织团队方向发展。

在我看来，管理者应该是一个教练的角色，更多的是协助团队成功，毕竟，团队的成功才是你的成功。

时间管理的诀窍

同样的工作，有些人可以有条不紊、井井有条，高效地完成；而另一些人则整天看起来很忙的样子，工作却迟迟不能按时收工。从一定程度上说，这是一个“时间管理”的问题。本话题邀请了在时间管理方面有话要讲的嘉宾，分享他们时间管理的诀窍。



王寒
云创新科技首席运营官

时间管理的七点体会

关于时间管理，不想拾人牙慧Copy Paste一大堆，只想谈一些个人的实际体会。

明确什么是自己“To Be”的目标

很认同永锡的一句话：时间管理的目的是让自己成为一个“幸福行动家”，是从“To Do”到“To Be”的一个必经过程。

很多时间管理理论都提到过，要把事情按重要紧急的程度来分类，但很少有人清楚该如何确定分类。有了“To Be”的定位，再根据自己的实际情况来制订中短期目标，就很容易知道该如何给具体的事情分类。

时间碎片化管理

对于现代人来说，时间被无穷无尽的杂事分割成了无数个小的碎片。 大多数人都处在“To Do”状态，因此需要主动把自己的工作时间分割成小的时间碎片。我曾尝试使用一个小时作为时间碎片单位，后来发现在办公室工作时基本不太可能。后来调整为每半小时一个时间碎片，每个时间碎片要完成某一件工作，或实现某个阶段性的成果。当然，在分割时间碎片时，要考虑自己的工作性质、工作环境、上司和团队成员的做事风格等因素。

此外，时间碎片的长度并非固定不变，要根据实际情况灵活调整。但一旦设定了当前的时间碎片，从开始的那一刻就要全力以赴。

最后，时间碎片可以和场景结合，比如有些事情要出去办，那么最好可以一次出去全部搞定，即便不能全部搞定，也应该取得预定的阶段性成果。

总之，管理时间碎片的原则是：宁愿主动分割，不要被迫分割。

排除工作中的干扰

最难管理的、同时也最不受自己控制的是工作时间中的打扰。

我的做法是：在办公室工作时间，不使用一切即时通信工具，每小时定期看一次是否有电话、短信、QQ。具体来说，首先把手机调成静音，然后是QQ、MSN这类的即时聊天工具，始终设置成隐身或离线状态。尽量使用邮件、微博这种离散通信工具。

排除工作中的干扰，还可以借助头戴式耳机，让自己在音乐中进入沉浸状态。

当然，如果有可能，应该和Boss以及团队成员就时间安排的问题多沟通，争取他们的理解和支持，尽量给自己提供独立完整的时间段，避免随时随地受到干扰。

了解自己的精力曲线

每个人都有自己的精力曲线，需要知道自己什么时间工作效率最高，从而把最难的事情分配到这个时间段的碎片中，而在自己昏昏欲睡、精力不济时可以做点最简单的事情。

随身带上纸和笔

我曾用过很长时间的Things，还试过OmniFocus和其他几款时间管理软件，但现在只保留Things作为备忘录来使用。

目前使用最频繁的时间管理工具，还是纸和笔。我习惯用永锡的方法，在纸上画上两

条线组成四个象限，分别代表重要且紧急、重要不紧急、不重要但紧急、不重要也不紧急。

不要花时间find my stuff

其实这个是我的弱项，写上来是要提醒大家，我们每天花在找各种小东西上的时间太多了。比如要吃饭时找钱包、要打电话时找手机、要写东西时找纸和笔。让一切井然有序可以大大节省这方面浪费的时间。

定期总结“To Do”和“To Be”

根据自己“To Be”的目标，定期来总结一下时间管理和自我的进步，无需沮丧，无需自傲，只要每次总结都有点小进步就OK了。

总之，时间管理的目标就是，从当前的“To Do”出发，最终达到“To Be”的状态。让自己来主动管理时间，而不是被时间管理。



何海涛
思科高级软件工程师

善用时间，发展副业

过去一年，我在保证不能影响本职工作的前提下，完成了《剑指Offer——名企面试官精讲典型编程题》一书的写作。下面来分享一下写作过程中践行的时间管理经验。

明确任务与目标

从2010年下定决心写书开始，我就设定了比较明确的目标。面试类的书籍有其特殊的时效性，最佳时间是在今年7月交稿，恰好能赶上2011年秋季的校园招聘。明确了任务与目标后，行动就有了方向。

在制订目标时，需要注意可操控性，除了计算正常工作量，一定需要保证适量的缓冲时间。毕竟只是副业，无法与本职工作有同等的优先级。目标如果过于激进，在突发状况发生时，进度就很难掌控。一般情况下，建议预留全部工作时间的25%~35%作为缓冲。

细化目标，执行计划

除了整体的计划外，还需要进行目标细化。在写书时，我不仅会进行每月的计划，同时会有每周甚至具体到每天完成多少内容的目标。计划的粒度越细，也越容易掌控。当某一天不能按时完成计划的工作量时，马上就能作出调整，而不至于到最后漏洞大了才恍然大悟。同时，写书是一个持久战，最考验人的不一定是文字能力，更多的是毅力。一个个小目标的实现，都是自我鼓励的基础，也是自信心的来源。

制订计划之后就要严格执行计划，尽量要求自己完成计划的工作量。

善用闲散时间

之前曾在网上看到一句话，说一个人的成就很大程度上决定于每天晚上8点到10点他在做什么。这是很多人晚餐之后看电视或者上网的时间。如果自己想在工作之余做出点成绩，就可以把这个休闲时间好好利用起来。

当然人总是需要一定的休息时间的，就比如一块肥沃的土地，连续几季都种同样的作物，土壤的肥力也会枯竭，收成就会越来越差。为解决这个问题，农民在土地上轮换种植不同的作物，或者同时种植几种作物，这样能提高土地的使用效率。其实时间也类似。写作之余我会到各论坛看看有没有关于程序员面试话题的讨论。这样一边休闲，一边为写书积累素材，休闲和正事两不误。

区分使用小段时间和大段时间

大脑思维在切换工作时需要一定的过程，并不能立即高效率地进入深度思考与完全工作状态。一般而言大脑需要10~15分钟才能集中精神进入工作状态，因此不宜用小段时间来做只有静下心来认真思考才能做好的工作。

如果空余时间不足1小时，我会选择把这段时间用来整理、修改已写好的书稿。这样即

使大脑没有完全集中精神也能开展。通常我会为撰写新内容安排一大段时间，从而避免大脑频繁切换工作场景，提高用脑效率。

小 结

一旦下定决心完成某一任务，就要在确定总体目标之后细化每一周甚至每一天的工作量，尽量确保每个时间节点都能按时完成。由于副业挤占了休息时间，可能会很累。虽然不是说完全不能有休闲活动，但如果有可能还是可以把闲散时间利用起来，做一些工作相关的事情，这样既放松了大脑，又有效利用了时间。如果能坚持一段时间，就会小有所成。



李铮
高级程序员、系统分析师

舍末逐本谈时间管理

时间管理是一个直观而复杂的概念，既可以只面向个人，也可以针对整个项目团队；既可以短暂应用于某一具体工作，也可以长期伴随整个企业的运转。因此，由这一概念衍生的讨论、技术、工具乃至理论纷繁浩帙不胜枚举。不过实施时间管理的基本策略大都一致，都会强调计划性和纪律性。然而，如果不认识到管理规则和计划并非一经拟定不容变更，就有可能陷入对时间管理形式的过分注重而忽略其本质——效率。下面结合亲身体验，尝试用三种截然不同的情形说明时间管理的本质在于力求效率而非机械地遵循“时间”安排。

改变时间以换取效率：在正确的时间做正确的事

我目前从事一些科研工作，撰写论文逐渐成为重心。学校官方的指导建议是坚持每天在固定时间段及固定的工作环境中。这在思路顺畅的情况下当然是大有裨益的，然而思路一旦卡壳，墨守成规地枯坐在电脑屏幕前往往更加一筹莫展。此刻不妨出门散步或是做一些其他无关的事情，等思路清晰的时候再回到工作环境，往往会事半功倍。制订并严格执行计划并不一定能保证工作的高效完成，视具体情况做出调整也应当成为时间管理的一部分。

追加时间以换取效率：磨刀不误砍柴工

我初入大学时计算机还是奢侈品，只有每周两节的上机课可以保证人手一台没硬盘的台式机以供练习编写C程序。一开始经常出现的一幕是还没等程序调通就下课了，无论如何有效利用上课时间也完成不了几道练习题。于是大家开始习惯于提前用纸和笔编程，然后带着程序草稿上机，迅速录入程序之后就有大把的时间进行调试了，练习的完成率也随之大大提高。类似的，对于时间集约型的工作而言，增加准备时间往往会带来更为可观的收效。

减少时间以换取效率：流程改进

重返校园之前，我在一家软件外包公司工作。在某产品准备发布的前期，美方开始不太公平地要求加班，很多工作要熬到前半夜等对方上班后交接处理，凌晨下班就成了家常便饭。经过一段时间的磨合，我们把最耗时的Bug重现工作不仅用文字和截图保留现场，而且用录像的方式把操作过程也保留下来，提前发给美方。这样一来，在交接时就只需短暂通话和少量协作即可，而不必再陪同演示了，下班时间也大为提前。换句话说，由于客观条件的制约，如果遵从既有的工作流程，很难想象能够优化时间安排为工作效率带来质的提高。而在工作量基本不变的情况下，通过改变工作流程来减少工作时间，自然也就达到了提高效率的目的。

结束语

时间管理的本质是通过规划提高时间使用效率，核心在于效率而非时间。虽然时间管理必然意味着针对时间的规则和约束，但如果忽视环境的改变或未能着眼于效率，即便博采制订计划、开列清单、诉诸工具等诸多手段，也会变得舍本逐末流于形式而收效甚微。

软件企业量化管理的思想方法

文 / 林锐

作者从事了十多年的软件企业研发管理方法的研究和实践，在他看来，软件企业是可以量化管理的。本文即是他对软件企业量化管理思想方法的全面介绍。

CMMI量化管理的思想方法

软件企业管理水平与CMMI等级之间的关系

我曾经和某中型软件企业的老总交流企业管理方法，这位老总没有读过CMMI相关书籍，他自己把企业管理水平分为4个级别。

第一级，新老员工能正常干活，能做项目。

第二级，公司各部门都有流程，所有人员都按照流程办事，否则人越多就越乱。

第三级，要有统计报表，让我看得明白大家做得怎么样，不要让我一个一个问过去。

第四级，发现不好的，要设法改进。发现好的，要发扬光大。

我接触的其他企业领导也表达了类似观点，上述4个级别与CMMI的2-3-4-5级恰好对应。

CMMI将企业能力成熟度分为5个级别：初始级、已管理级、已定义级、量化管理级、优化级，如图1所示。

CMMI的5个成熟度等级为评价软件企业能力提供了一个有序的级别，同时也为软件过程改进工作指明了方向，让人们分清轻重缓急，指导人们一步一步地提升过程能力而不是企图跳跃式地前进。

CMMI 2-3-4-5级的量化管理逻辑

看了CMMI的成熟度等级划分，千万不要误解只有到了CMMI4级才搞量化管理！

CMMI2级有个过程域叫“度量与分析”，可见从CMMI2级就开始搞量化管理了。所以说，CMMI从2级到5级，每一级都有量化管理，只是范围和程度不一样而已。

CMMI 2-3-4-5级量化管理关系如图2所示。

■ CMMI2级量化管理关注“项目级过程实践的度量与分析”。

■ CMMI3级量化管理关注“组织级过程实践的度量与分析”。

■ CMMI4&5级量化管理是围绕企业目标、基于过程绩效模型的度量与分析。

CMMI4&5级共4个过程域，它们之间相互关联，不要孤立使用。即使企业正在实施CMMI4级，也不能忽略CMMI5级的2个过程域。CMMI4&5级量化管理的结果，最终会落实到改进CMMI2&3级的过程与实践。

企业员工们的大部分工作都在CMMI2&3级的过程域范围之内，CMMI4&5级量化管理不仅没有给员工增加额外的工作负担，反而可以指导改进员工们的日常工作。



图1 CMMI的5个成熟度等级



图2 CMMI2-3-4-5级量化管理关系图

软件企业如何开展量化管理

软件企业的量化管理实施模型



图3 量化管理实施模型

软件企业的量化管理实施模型如图3所示。

第1步：设定和调整组织目标。

组织目标分两类：商业目标和过程能力目标。商业目标比较直观，相对容易设定，例如多少人员、生产多少东西、卖多少钱。而过程能力目标则比较抽象，不容易设定，例如研发能力、客服能力，它和很多因素相关，不是一句话、几个数字就能说得明白的。

刚开始时，大家并不清楚过程能力目标倒底是多少，无从下手。一般地，要先设计“过程绩效模型”，用一组度量元来反映过程能力。对企业当前情况经过统计分析后，获得当前的基准值（CMMI称之为过程绩效基线）。基准值反映了某过程当前的真实能力水平，可以把这个基准值设定为过程能力目标的初值，将来再调整。

商业目标和过程能力之间的关系：企业能否实现既定的商业目标，与其过程能力很有关系。一般地，过程能力越强，越容易实现商业目标。但实现商业目标并不完全取决于过程能力，社会环境、机遇、偶然因素都会影响目标的实现。一群高水平的人不见得就能把公司经营好。所以过程能力是实现商业目标的必要条件，而非充分条件。从企业管理的角度讲，应当下功夫提升过程能力，把基础打好，必然有利于实现商业目标。

为什么要花精力设定目标和调整目标呢？中国流传“吹牛皮不犯法”，企业随便设定高一些的目标，给外人看看行不行？

不行！如果企业是上市公司，有意吹牛皮是犯法的（误导股民、扰乱股市）。即使没有遭到法律惩罚，也会失信于客户和员工，最终危害企业自身。

第2步：制订流程制度，设计度量元和统计方法。

EPG（过程工程组）负责制订和优化流程制度。如果流程不完整，则要查漏补缺。若已有流程存在不合理之处，则要改进流程。并为每个流程设计度量元及其统计方法。

这个步骤的目的是：建立标准化的流程制度，使人们“有法可依”，并且用度量元来反映流程执行的好坏程度。

第3步：对员工进行流程和技能培训。

首先对员工进行流程培训，让员工们知道应该怎样开展工作。其中有些流程存在一定的技术难度，例如设计，如果员工的技能较差，即使他知道设计流程，也做不出好的设计成果。这个时候，还要对员工进行较长时间的技能培训。

注意：对于软件企业而言，技能培训比流程培训难度更大、消耗精力更多。员工技能的提升不可能速成，技能培训是一项持久的工作。

第4步：真实过程的条目化记录。

这个步骤有三个要素。

■ 及时记录。例如测试过程发现缺陷，要马上记录缺陷。否则时间长了就记不起来了，而且会耽误其他人的工作。

■ 真实信息。记录真实情况，绝对不可以造假，也不鼓励事后修改过去不完美的痕迹。例如项目任务进度表，要输入真实的计划和真实的执行情况，即使偏差很大，它也是有价值的，是宝贵的过程信息。不可篡改执行情况（这是造假），也不要修改计划来迎合实际执行情况（这是马后炮，造成了计划很完美的假象，无实用价值，却使统计分析失真）。

■ 条目化。在记录工作情况时，要设法“条目化”，即把内容分拆成若干条，每一条都有负责人、时间、状态。便于跟踪处理和统计分析。

第5步：统计分析，获得量化结果。

按照预先设计的度量元和统计方法，对大量的条目化记录（源数据）进行统计分析，计算出度量元的值，并且按预先设计的格式产生报表。

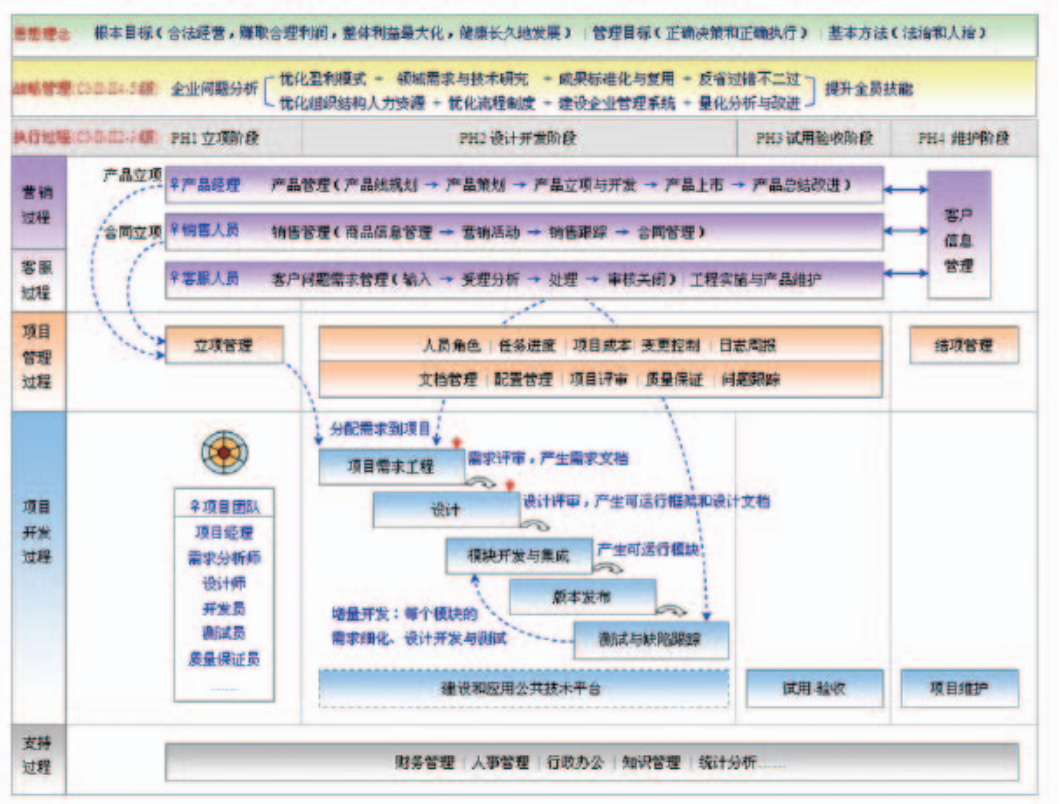


图4 软件企业的集成化流程IDP

注意：第4步和第5步，一定要使用配套的软件工具。仅靠手工处理数据和统计分析，枯燥乏味且工作量极大，几乎没有人坚持得住。

第6步：对比目标和结果，分析原因，给出对策

对比“量化结果”和“组织设定的目标”，如果两者存在比较大的差异，则要分析根本原因，然后给出解决问题和优化的方案，并且执行这些方案。

经过第1步至第6步的量化管理活动，最终使组织目标更加切合实际情况，目标可管理，企业持续进步。

软件企业的集成化流程

易于执行、并且能为企业创造效益的流程制度，不仅是量化管理的基础，而且它本身就是企业的财富。

自CMM和CMMI从2000年左右引入中国后很快得以普及。国内有不少流程改进社团，很多软件企业内部也有流程改进小组，大家熟知流程改进的概念。

很多软件企业基于CMMI研制流程，推行流程，原本是件好事。但令人遗憾的是，我见过的大部分流程都是围绕“CMMI认证”而非“围绕企业盈利”而制订的，严重缺乏实用性。

从2000年起我开始创作切合国内软件企业的集成化流程（IDP），如图4所示。并且开发了配套了的软件工具“集成化研发管理平台MainSoft”。经过十多年的实践和改进，IDP已相当成熟，国内有数十万读者，有一定的普适性。

IDP具有如下特点。

■ 把软件企业的主要过程“营销过程、客服过程、项目管理过程、项目开发过程、支持过程”集成一起，使企业流程一目了然，便于全员有条不紊地开展工作。

■ 以企业盈利为目标，既精简了CMMI2-3-4-5级过于庞大复杂的条款，又补充了CMMI不涉及的营销、客服过程实践，符合企业实际需求。

■ IDP流程有配套的软件工具“集成化研发管理平台MainSoft”，企业根据自身情况裁剪IDP流程和MainSoft功能，容易实现“规范化、量化”的研发管理。

常用的算术统计

求和

求“和”一般有两种方式：

■ 计算总和，即所有样本值的累加，例如年销售额，是一年内所有销售记录的总和。

■ 分段求和，例如一年分12个月，计算每个月的销售额，共有12个分段求和。

均值

“均值”等于“样本值之和”除以“样本数量”，例如“月销售额均值=年销售额/12*100%”。

均值计算简单，被人们普遍使用。但当样本之间差异很大时，均值会掩盖样本差异，失去参考价值（或产生误导）。例如把“富人和穷人”放在一起求均值（平均财富、平均消费等），就会掩盖贫富差异的真相，误以为没有穷人。

当样本之间差异很大时，把所有样本放在一起求均值意义不大。要设法把样本分段，使每段内的样本差异比较小，再计算每段的均值。

百分比

计算每个类别样本占总数的百分比，用于反映样本的分布规律，便于企业及时了解状况，调整资源部署。

增长率

增长率=（当前值–过去值）/过去值*100%，常用的是年增长率和月增长率。

■ 年增长率，如年销售额增长率=（今年销售额-去年销售额）/去年销售额*100%。

■ 月增长率又分两种：环比和同比。环比是本月和上月相比，即（本月值-上月值）/上月值*100%。而同比是本月和去年本月相比，即（本月值-去年本月值）/去年本月值*100%。

常用的统计图

绘制统计图的好处是：直观、醒目地展示数据，加深人们对统计数据的印象和洞察力。

饼图

饼图（Pie Chart）用于显示每个类别样本占总数的百分比。

柱状图和条状图

柱状图（Column Chart）和条状图（Bar Chart）都用于显示样本的数值，例如显示12个月的销售额，x轴表示月份，y轴表示销售额。

柱状图的x轴是横向的（向右延伸），y轴是竖向的。当样本数量不多时，柱状图符合人们的阅读习惯。当样本数量很多时，样本难以在一个页面内显示，要左右滚动页面，阅读不方便。

条状图的x轴是竖向的（向下延伸），y轴是横向的，适合样本数量比较多的情况。当一个页面显示不下时，可以上线滚动页面。

频率直方图

频率直方图（Frequency Histogram）将某变量的数值范围等分为若干区间，统计该变量在各个区间上出现的频率，并用矩形条的长度表示频率的大小。利用频率直方图可以分析数值的分布特征。

频率直方图常用于质量分析，判断生产过程是否处在正常状态（正态分布）。正态分布直方图的形状是中间高、两边低，左右近似对称，如图5所示。正常生产条件下，如果得到的直方图不是正态形状，就要分析原因，采取相应措施。

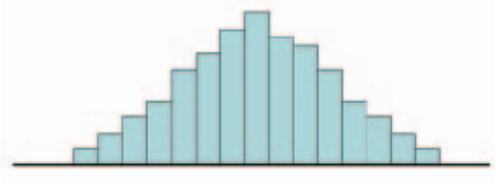


图5 正态分布的直方图

除了质量分析之外，频率直方图在营销统计分析也很有用处。

制作直方图最关键的是对样本进行分组，调整“分组数”和“组距”将改变直方图的现状。通常要多试验几次，才能找出反映分布规律的直方图。

控制图

控制图是根据假设检验的原理构造一种图，用于监测生产过程是否处于控制状态。它是统计质量管理的一种重要手段和工具。

在生产过程中，产品质量由于受随机因素和系统因素的影响而产生偏差；前者由大量微小的偶然因素叠加而成，后者则是由可辨识的、作用明显的原因所引起，经采取适当措施可以发现和排除。当一生产过程仅受随机因素的影响，从而产品的质量特征的平均值和偏差都基本保持稳

定时，称之为处于控制状态。此时，产品的质量特征是服从确定概率分布的随机变量，它的分布（或其中的未知参数）可依据较长时期在稳定状态下取得的观测数据用统计方法进行估计。分布确定以后，质量特征的数学模型随之确定。为检验其后的生产过程是否也处于控制状态，就需要检验上述质量特征是否符合这种数学模型。为此，每隔一定时间，在生产线上抽取一个大小固定的样本，计算其质量特征，若其数值符合这种数学模型，就认为生产过程正常，否则，就认为生产中出现某种系统性变化，或者说过程失去控制。这时，就需要考虑采取包括停产检查在内的各种措施，以期查明原因并将其排除，以恢复正常生产，不使失控状态延续而发展下去。

应用最广的控制图是W.A.休哈特在1925年提出的，一般称之为休哈特控制图，如图6所示。它的基本结构是在直角坐标系中画三条平行于横轴的直线，中间一条实线为中线，上、下两条虚线分别为上、下控制界限。横轴表示按一定时间间隔 抽取样本的次序，纵轴表示根据样本计算的、表达某种质量特征的统计量的数值，由相继取得的样本算出的结果，在图上标为一连串的点子，它们可以用线段连接起来。

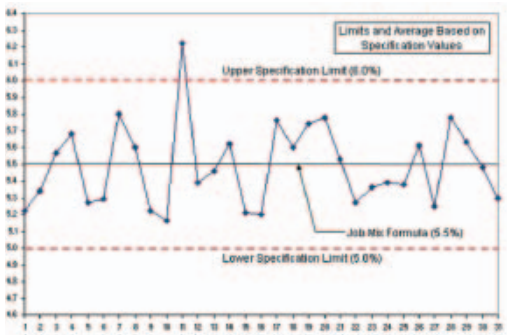


图6　控制图示例

中间的一条为中心线（Central Line，CL），一般用蓝色的实线绘制；在上方的一条称为控制上限（Upper Control Limit，UCL）；在下方的称为控制下限（Lower Control Limit，LCL）。对上、下控制界限的绘制，则一般均用红色的虚线表现，以表示可接受的变异范围，至于实际产品质量特性的点连线条则大多用黑色实线绘制。

统计过程控制（SPC）中最常用的控制图是遵循正态分布计量特征的平均数—极差（X-R）控制图，如图7所示。

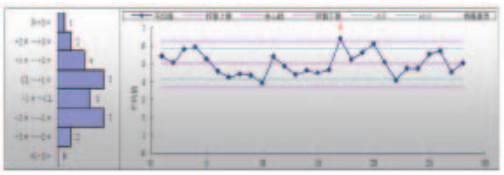


图7　平均数—极差（X-R）控制图示例

回归分析介绍

变量之间的关系一般分为两种：一种是完全确定关系，即函数关系；一种是相关关系，即变量之间既存在着密切联系，但又不能由一个或多个变量的值求出另一个变量的值。

对于彼此联系比较紧密的变量，人们总希望建立一定的公式，以便变量之间互相推测。回归分析的任务就是用数学表达式来描述相关变量之间的关系。例如，凭经验我们知道“产品质量”和“用户满意度”变量密切相关，但这两个变量之间到底是哪个变量受哪个变量的影响，影响程度如何，则需要通过回归分析方法来确定。

回归分析按照涉及的自变量的多少，可分为一元回归分析和多元回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。如果在回归分析中，只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为多元线性回归分析。

一般来说，回归分析是通过规定因变量和自变量来确定变量之间的因果关系，建立回归模型，并根据实测数据来求解模型的各个参数，然后评价回归模型是否能够很好地拟合实测数据；如果能够很好地拟合，则可以根据自变量作进一步预测。

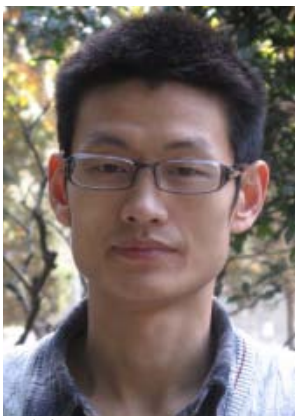


林锐

上海漫索科技创始人兼总经理，致力于创作适合国内企业需求的研发管理方法论和软件产品。累计出版《软件工程与项目管理解析》、《高质量程序设计指南——C++/C语言》等8部图书。

责任编辑：董世晓（dongsx@csdn.net）

云时代的职业挑战



杨海朝
新浪首席DBA

技术人员需要多才多艺

2011年是云计算风起云涌的一年，云计算在国内开始走下云端，落地应用。它会最终促使全球资源集中化，各种资源通过各种技术按照市场的规则被归类集中起来，从而使得资源的利用效达到最大化。

云计算会引起人们思维方式发生很大的改变，让我们学着从整体的角度来应对各种问题，用均衡的思想去处理整个系统里局部之间的关系。它对现有的软硬件厂商带来全新的创业和创新机会，要想抓住云计算，就需要把产品功能不断完善，去抓住云计算的机会。

云计算涉及的分布式、虚拟化和自动化管理等技术均不是新技术，硬件方面也是大家所熟知的产品。可以说，云计算并不是技术的创新，而是让老技术通过重新组合发挥出更大的商业价值，是一种商业模式的创新。它的市场规模巨大。以后软件慢慢地将会以服务的形式来提供，服务将通过互联网来实现。软件是服务的载体，服务是软件的表现形式，软件和服务之间的边界会慢慢变得模糊。云计算的兴起使得传统的软件外包将发生变化，外包的重心将从流程转向软件服务。

云计算不仅影响着计算机技术的发展，还对IT专业人士的就业产生了重大影响。云计算会创造一些新的就业岗位，即便是所有的后台管理和维护工作都交给了云供应商，企业也需要对基础设施建设有比较深刻的掌握和理解。云计算催生了对这方面人才需求的同时，作为IT技术人如何在这股大潮中脱颖而出？如果想要在云计算开辟的新环境下获得成功，技术人员需要多才多艺，而且精通各种技术。并且需要与其他非技术部门共事，需要每一个人都能身兼数职。

我们需要多学习分布式理论和工程实践以及学习安全方面的知识和法规，编写软件慢慢向服务转变，能解决大规模并行计算、数据分布存储、数据实时备份、应用高度集成以及安全可靠和个性化应用等问题。能将公司的内部基础实施和外部基础设施相衔接的混合技术是很受青睐的。



孙立
去哪儿网高级系统架构师

学习更多的云计算开发模式

话说云计算现在在概念上还不是特别的清晰，有很多厂商还是在炒作中。我相信云计算的确会给整个IT带来了很大的震动，但这种对行业的震动还得分国外和国内。

国外的确有很多公司把自己的IT系统部署到Amazon上，比如Netflix，节省了大量的运维人力成本。

但国内来说，由于众所周知的政策原因，把IT系统托管到第三方，总会觉得不安全，所以像国外那样大量使用Amazon的情况并不常见。所以对国内的就业影响目前还不是很明显。但从长远来看，可能对某些IT外包系统会有些影响。

对于职业来说，云计算首先影响的人可能会是IT管理员和系统运维人员。但同样，云计算公司会需要大量的更加高级的IT管理员和运

维人员。而那些基础管理员和运维人员需求将变少。

对于开发人员，可能需要学习更多的云计算开发模式。这跟以前用ASP开发网页，到现在都基本没市场了一样，但这些人并没有失业，只是转入到更加高级的开发中去了。

作为一名架构师，我也想谈谈架构师这个职业可能面临的风险。

费用问题。虽然大部分宣传和文章都在讲云计算如何便宜，但云计算并不能帮助架构师解决太多的问题。在云计算面前，同样需要根据云计算的特性去设计系统，以应对业务需求，用最少的资源满足最大的业务。虽然云计算能够很好地扩展，但还是要保证系统的性能，不然那笔客观的费用还不如不用云计算呢。

安全问题。云计算提供商并没有保证100%的高可用性，所以你还得准备应对高可用问题。对于数据本身的安全性，要保证不被厂商泄密。

迁移问题。当你选择一家云计算服务厂商，一旦投入使用，你是否还可以迁移到另外一家，成本又是多高？如果被一家捆死，那将是一个噩梦。我相信云计算厂商都希望把客户捆死的。

国内的政策监管。云计算厂商会不会一夜之间就被关闭，让你遭受灭顶之灾。

赔偿问题。如果出现重大问题，损失的赔偿问题。



吴朱华
人云科技创始人

要形成自己的核心竞争力

在之前的PC时代和互联网时代，人们都有一些顾虑，觉得智能化新技术的到来和采用将会导致就业人数急剧减少。

但实际上，无论是PC还是互联网这样新技术的到来，其实都对就业有极大的促进作用，其中最明显的例子，莫过于电子商务。虽然现在的电子商务对实际线下的消费有所影响，但它本身所需的一整套产业链，从网站展示到支付和物流等都需要大量的从业人员，而且电子商务这种新模式也引起更多的消费。而且以美国为例，现阶段IT方面的就业情况和很多传统行业相比，应该光明很多。

首先，在我看来，新的技术将有力地促进就业：

其一，由于云计算技术到来，使得很多过去认为很艰难的事情成为可能，比如海量数据分析、物联网和移动互联网等，而这些新的发展将会导致更多专业人士得到工作；

其二，新的大型云计算中心建设和云计算相关产业链将会产生越来越多新的职位，也需要更多的专业人士投入其中。

其次，我认识云计算能使职业发展更专业化，也主要有下面这两点：

其一，由于公有云的越来越常见，将会减少那些在低效率的企业内部数据中心工作的系统管理人员，而在那些高效率的大型云计算中心则会增加一些系统管理员，而且这些系统管理员将会在大型云计算中心这样的大舞台得到更大的发挥和培养，以及长期经验的积累；

其二，由于云计算将会提供诸如PaaS这样高效率的开发模式，将使得很多企业的内部IT人员将会慢慢地将更多关注力从技术转向业务。这不仅能帮助这些IT人员得到更好的职业发展，而且将帮助企业的业务更好地与IT服务之间进行嫁接。

最后，我想和大家说，不要惧怕新浪潮的到来。因为新浪潮并不使你过去的经验一文不值，而是去适应这股新的浪潮，学习一些新的东西，并且在平时的实践中进行不断的积累，从而形成自己的核心竞争力，比如，对技术或者业务深刻的理解，也可以是人脉等。

如何“打败”CAP定理

文 / Nathan Marz　译 / 方建

CAP定理是数据系统设计的基本理论，目前几乎所有的数据系统的设计都遵循了这个定理。但CAP定理给目前的数据系统带来了许多复杂的、不可控的问题，使得数据系统的设计越来越复杂。Twitter首席工程师、Storm的作者Nathan Marz在本文中通过避开CAP定理带来的诸多复杂问题，展示了一个不同于以往的数据系统设计方案，给我们的数据系统设计带来了全新的思路。

CAP定理指出，一个数据库不可能同时满足一致性（Consistency）、可用性（Availability）和分区容错性（Partition-Tolerance）。

一致性（Consistency）是指执行了一次成功的写操作之后，未来的读操作一定可以读到这个写入的值。可用性（Availability）是指系统总是可读可写的。Yammer的Coda Hale和Cloudera的Henry Robinson都阐述过，分区容错性是不能牺牲的，因此只能在一致性和可用性上做取舍，如何处理这种取舍正是目前NoSQL数据库的核心焦点。

选择一致性而不是可用性的系统将面临一些尴尬的问题，当系统不可用时怎么办？你可以对写操作进行缓冲处理，但如果存储缓冲数据的机器出现故障，客户端将丢失写入的值。同样地，缓冲写也可以被认为是一种非一致性的操作，因为客户端认为成功的写入实际上并没有写入到实际的数据库中。当然，系统可以在机器不可用时向客户端返回错误，但可以想象，一个经常告诉客户端“请重试”的产品是多么令人讨厌。

另一个方案是选择可用性放弃一致性。这种情况下最好的一致性保障是“最终一致性”（Eventually Consistency）。当使用最终一致性的系统时，客户端有时会读到与刚刚写入数据不同的数据。有时候，同一时间同一个key的多个请求有可能返回不同的结果。数据更新并不能及时在所有的复制节点上生效，所以不同的复制节点上可能读取到的是不同的值。当你检测到数据不一致性时，你需要进行修复（Repair）操作，这就需要使用矢量时钟（vector clock）记录数据的版本历史并合并不同的数据更新（这称为读取修复，read repair）。

我相信在应用层维护最终一致性对开发人员负担太重，开发人员极易弄错读取修复的代码，而一旦开发人员犯错，有问题的读取修复将对数据库系统造成不可逆的损坏。

所以牺牲可用性时问题会很多，牺牲一致性时构建和维护系统的复杂度又很高，但这里又只有两个选择，不管怎样做都会不完美。CAP定理是改不了的，那么还有什么其他可能的选择吗？

实际上，还有一个办法：你并不能避开CAP定理，但可以把复杂的问题独立出来，免得你丧失对整个系统的掌控能力。CAP定理带来的复杂性，其实是我们如何构建数据系统这一根本问题的体现。其中有两点特别重要：数据库中可变状态和更新状态的增量算法。复杂性正是这两点和CAP定理之间的相互作用导致的。

本文将通过一个数据库系统的设计，来说明如何解决CAP定理通常会造成的复杂性问题。但我要做的不仅仅如此，CAP定理是一个针对机器发生错误时系统容错性的一个定理，而这里有比机器容错性更加重要的容错性——人为操作容错性。在软件开发中一个确定的事实是，开发人员都并非完人，产品中难免有一些Bug，我们的系统必须对有Bug的程序写入的错误数据有足够的适应能力，我要展示的系统将是这样一个可以容忍人为错误的系统。

本文将挑战你对数据系统如何构建这一问题的假设，通过颠覆传统数据系统构建方法，我会让大家看到一个前所未见的优雅、扩展性强、健壮的数据系统。

什么是数据系统？

在开始介绍系统设计之前，让我们先来看看

我们要解决的问题：数据系统的目的在于什么？什么是数据？ 在我们考虑CAP定理之前，我们必须给出一个可以适用于所有数据应用程序的定义来回答上述问题。

数据应用程序种类很多，包括存入和提取数据对象、连接、聚合、流处理、机器学习等。似乎并不存在一个对数据系统的明确定义，数据处理的多样性使得我们很难用一个定义来描述。

事实却并非如此，下面这个简单的定义：

Query = Function(All Data)

概括了数据库和数据系统的所有领域。每一个领域——有50年历史的RDBMS、索引、OLAP、OLTP、MapReduce、EFL、分布式文件系统、流处理器、NoSQL等——都可以被概括进这个方程。

所谓数据系统就是要回答数据集问题的系统，这些问题我们称之为“查询”。上面的方程表明，查询就是数据上的一个函数。

上述方程对于实际使用来说太过于笼统，几乎对复杂的数据系统设计不起什么作用。但如果所有的数据系统都遵循这个方程又会怎样呢？这个方程是探索我们数据系统的第一步，而它最终将引导我们找到“打败”CAP定理的方法。

这个方程里面有两个关键概念：数据、查询。这两个完全不同的概念经常被混为一谈，所以下面来看看这两个概念究竟是什么意思。

数据

我们先从“数据”开始。所谓数据就是一个不可分割的单位，它肯定存在，就跟数学里面的公理一样。

关于“数据”有两个关键的性质。首先，数据是跟时间相关的，一个真实的数据一定是在某个时间点存在于那儿。比如，假如Sally在她的社交网络个人资料中写她住在芝加哥，你拿到的这个数据肯定是她某个时间在芝加哥填写的。假如某天Sally把她资料里面居住地点更新为亚特兰大，那么她肯定在这个时候是住在亚特兰大的，但她住在亚特兰大的事实无法改变她曾经住在芝加哥这个事实——这两个数据都是真实的。

其次，数据无法改变。由于数据跟某个时间点相关，所以数据的真实性是无法改变的。没有人可以回到那个时间去改变数据的真实性，这说明了对数据操作只有两种：读取已存在的数据和添加更多的新数据。那么CRUD就变成了CR【译者注：CRUD是指Create Read Update Delete，即数据的创建、读取、更新和删除】。

我去掉了“更新”操作，因为更新对于不可改变的数据没有任何作用。例如，更新Sally的位置信息本质上就是在她住的地方数据中新加一条最近的位置信息而已。

我同样去掉了“删除”操作，因为绝大部分删除操作可以更好地表述为新加一条数据。比如Bob在Twitter上不再关注Mary了，这并不能改变他曾经关注过Mary这个事实。所以与其删除Bob关注Mary这个数据，还不如新加一条Bob在某个时间点不再关注Mary这个数据。

这里只有很少数的情况需要永久“删除”数据，例如规则要求你每隔一段时间清掉数据，这个情况在我将要展示的系统中有很好的解决方案，所以为了简洁，我们暂不考虑这些情况。

查询

查询是一个针对数据集的推导，就像是一个数学里面的定理。例如，你可以通过计算“Sally现在的位置在哪里”这个查询来得到Sally最新的位置数据。查询是整个数据集合上的函数，可以做一切事情：聚合、连接不同类型的数据等。因此，你可以查询系统中女性用户的数量，可以查询最近几小时热门的Twitter内容。

前面我已经定义查询是整个数据集上的函数，当然，不是所有的查询都需要整个数据集，它们只需要数据集的一个子集。但我的定义是涵盖了所有的查询类型，如果想要“打败”CAP定理，我们需要能够处理所有的查询。

打败CAP定理

计算查询最简单的办法就是按照查询语义在整个数据集上运行一个函数。如果这可以满足你对延迟的要求，那么就没有其他需要构建的了。

可想而知，我们不能指望在整个数据集上的查询能够很快完成，特别是那些服务大型网站、需要每秒处理几百万次请求的系统。但假如这种查询可以很快完成，让我们来看看像这样的系统和CAP定理的PK结果：你将会看到，这个

系统不仅打败了CAP定理，而且还消灭了它。

CAP定理仍然适用，所以你需要在可用性和一致性上做出选择，这里的漂亮之处在于，一旦你权衡之后做出了选择，你就做完了所有的事情。通常的那些因为CAP定理带来的问题，都可以通过不可改变的数据和从原始数据中计算查询来规避。

如果你选择一致性而不是可用性，那么跟以前并没有多大的区别，因为你放弃了可用性，所以一些时候你将无法读取或者写入数据。当然这只是针对对强一致性有要求的系统。

如果你选择可用性而不是一致性，在这种情况下，系统可以达到最终一致性而且规避了所有最终一致性带来的复杂问题。由于系统总是可用的，所以你总可以写入新数据或者进行查询。在出错情况下，查询可能返回的不是最近写入的数据，但根据最终一致性，这个数据最终会一致，而查询函数最终会把这个数据计算进去。

这里的关键在于数据是不可变的。不可变数据意味着这里没有更新操作，所以不可能出现数据复制不同这种不一致的情况，也意味着不需要版本化的数据、矢量时钟或者读取修复。在一个查询场景中，一个数据只有存在或者不存在两种情况。这里只有数据和在数据之上的函数。这里没有需要你为确保最终一致性额外做的事情，最终一致性也不会因此使你的系统变得复杂。

之前的复杂度主要来自增量更新操作和CAP定理之间的矛盾，在最终一致性系统中可变的值需要通过读取修复来保证最终一致性。通过使用不可变数据，去掉增量更新，使用不可变数据，每次从原始数据计算查询，你可以规避那些复杂的问题。CAP定理就被打败了。

当然，现在讲的只不过是想法而已，而且每次从原始数据计算查询基本上不可能。但我们从中可以学到一些在实际解决方案中的关键点。

- 数据系统因为不可变数据和不断增长的数据集变得简单了。
- 基本的写入操作就是写入一条新的不可变数据。
- 数据系统通过重新从原始数据计算查询规避了CAP定理带来的复杂度。
- 数据系统利用增量算法使得查询的返回延迟降低到一个可以接受的程度。

让我们开始探索这个数据系统应该如何设计。请注意从这里开始我们所描述都是针对系统优化、数据库、索引、EFL、批量计算、流处理——这些技术都是对查询函数的优化，让查询返回时间降低到一个可以接受的程度。这很简单，但也是数据系统所面对的现实。数据库通常是数据管理的核心，但它们是更大蓝图中的一部分。

批量计算

“如何让任意一个函数可以在任意一个数据集上快速执行完成”这个问题太过于复杂，所以我们先放宽了一下这个问题依赖条件。首先假设，可以允许数据滞后几小时。放宽这个条件之后，我们可以得到一个简单、优雅、通用的数据系统构建解决方案。之后，我们会通过扩展这个解决方案使得它可以不用放宽条件来解决问题。

由于查询是所有数据的一个函数，让查询变快的最简单的方法就是预先计算好这些查询。只要这里有新的数据，你就重新计算这些查询。这是可能的，因为我们放宽了条件使得我们的数据可以滞后几个小时。图1展示了这个工作流程。

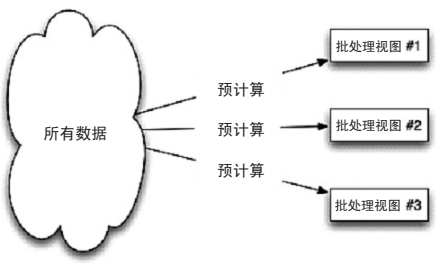


图1 预计算工作流程

为了实现这个，你的系统需要：

- 能很容易存储大的、不断增长的数据集；
- 能在数据集上可扩展地计算查询函数。

这样的系统是存在的，即Hadoop。它是一个成熟的、经历了无数团队实战检验过的系统，同时拥有一个巨大的工具生态系统。它虽不完美，但是这里用来做批量处理的最好的一个工具。

许多人也许会告诉你，Hadoop只适用于那些“非结构化”的数据，这是完全错误的看法。Hadoop处理“结构化”的数据也很不错，通过使用像Thrift或者Protocol Buffers这样的工具，你可

以使用丰富的数据结构存储你的数据。

Hadoop由分布式文件系统HDFS和批处理框架MapReduce两部分构成。HDFS可以通过文件存储大量数据，MapReduce可以在这样数据上进行可扩展计算。这个系统完全符合我们的要求。

我们将数据以文件形式存储到HDFS中去。文件可以包括一个数据记录序列。新增数据时，我们只需要在包括所有数据的文件夹中新增一个包含这条新记录的文件即可。像这样在HDFS存储数据满足了“能够很容易存储大的、不断增长的数据集”这个要求。

预计算数据集上的查询也很直观，MapReduce是一个足够复杂的框架，使得几乎所有的函数都可以按照多个MapReduce任务这种方式实现。像Cascalog、Cascading和Pig这样的工具使实现这些函数变得十分简单。

最后，为了可以快速访问这些预计算查询结果，你需要对查询结果进行索引，这里有许多数据库可以完成这个工作。ElephantDB和Voldemort read-only可以通过从Hadoop中导出key/value数据来加快查询速度。这些数据库支持批量写和随机读，同时不支持随机写。随机写使得数据库变得复杂，所以通过不支持随机写，这些数据库设计得特别简洁，也就几千行代码而已。简洁使得这些数据库鲁棒性变得非常好。

下面来看批量处理系统整体上是如何配合工作的。假设写一个网站分析程序来跟踪页面访问量，你需要能够查询到任意时间段的页面访问量，数据是以小时方式提供的。如图2所示。

实现这个很简单，每一个数据记录包括一个单一页面的访问量。这些数据通过文件形式存储到HDFS中，一个函数通过实现MapReduce计算任务，来计算一个URL下页面每小时的访问量。这个函数产生的是key/value对，其中[URL, hour]是key，value是页面的访问量。这些key/value对被导出到ElephantDB中去，使得应用程序可以快速得到任意[URL, hour]对对应的值。如果应用程序想要知道某个时间范围内某个页面的访问量，它可以查询ElephantDB中那段时间内的数据，然后把这些数据相加就可以得到这个访问量数据了。

在数据滞后几小时这个缺陷下，批量处理可以计算任意数据集上的任意函数。系统中的“任意性”是指这个系统可以处理任何问题。更重要的是，它很简单，容易理解和完全可扩展，你需要考虑的只是数据和查询函数，Hadoop会帮你处理并行的事情。

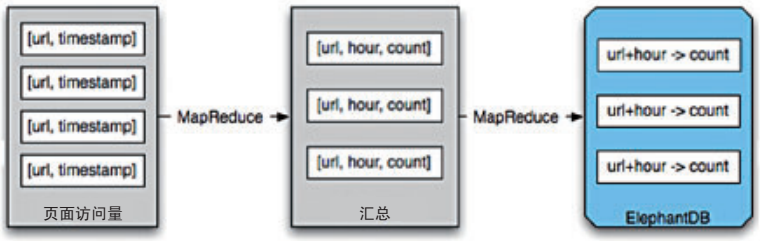


图2 批处理工程流程示例(timestamp代表时间戳，count代表个数)

批处理系统、CAP定理和容忍人为错误

截至目前，我们的系统都很不错，这个批处理系统是不是可以达到容忍人为错误的目标呢?

让我们从CAP定理开始。这个批处理系统总是最终一致的：写入的数据总可以在几小时后被查询到。这个系统是一个很容易掌控的最终一致性系统，使得你可以只用关注你的数据和针对数据的查询函数。这里没有涉及读取修复、并发和其他一些需要考虑的复杂问题。

接下来看看这个系统对人为错误的容忍性。在这个系统中人们可能会犯两个错误：部署了一个有Bug的查询函数或者写入了错误的数据。

如果部署了一个有Bug的查询函数，需要做的所有事情就是修正那个Bug，重新部署这个查询函数，然后在主数据集上重新计算它。这之所以能起作用是因为查询只是一个函数而已。

另外，错误的数据有明确的办法可以恢复：删除错误数据，然后重新计算查询。由于数据是不可变的，而且数据集只是往后添加新数据，写入错误的数据不会覆盖或者删除正确的数据，这与传统数据库更新一个数据就丢掉旧的数据形成了鲜明的对比。

注意到MVCC和HBase类似的行版本管理并不能达到上面人为错误容忍级别。MVCC和HBase行版本管理不能永久保存数据，一旦数据库合并了这些版本，旧的数据就会丢失。只有不可变数据系统能够保证你在写入错误数据时可以找到一个恢复数据的方法。

实时层

上面的批量处理系统几乎完全解决了在任意数据集上运行任意函数的实时性需求。任何超过几个小时的数据已经被计算进入了批处理视图中，所以剩下来要做的就是处理最近几个小时的数据。我们知道在最近几小时数据上进行查询比在整个数据集上查询要容易，这是关键点。

为了处理最近几个小时的数据，需要一个实时系统和批处理系统同时运行。这个实时系统在最近几个小时数据上预计算查询函数。要计算一个查询函数，需要查询批处理视图和实时视图，并把它们合并起来以得到最终的数据。

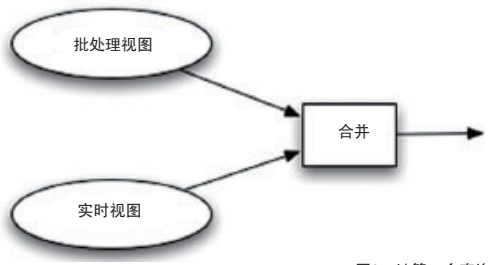


图3 计算一个查询

在实时层，可以使用Riak或者Cassandra这种读写数据库，而且实时层依赖那些数据库中对状态更新的增量算法。

让Hadoop模拟实时计算的工具是Storm。我写Storm的目的是让Hadoop可以健壮、可扩展地处理大量的实时数据。Storm在数据流上运行无限的计算，并且对这些数据处理提供了强有力的保障。

让我们回到刚才那个根据某个URL查询某个页面在某个时间段内页面访问量的例子，通过这个例子我将展示实时层是如何工作的。

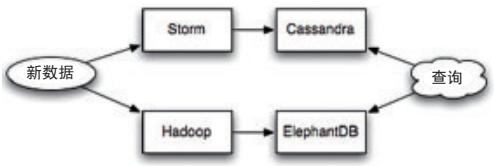


图4 批处理/实时架构示例

批处理系统还是跟之前一样：一个基于Hadoop和ElephantDB的批处理工作流，在几个小时之前的数据上预计算查询函数。剩下就是让实时系统去处理最近几小时数据了。

我们将最近几小时的数据状态存入Cassandra中，用Storm去处理页面访问量数据流并并行更新到数据库中，针对每一个页面访问量，在[URL, hour]所代表的key下，有一个计数器，这个计数器在Cassandra中实现。这就是所有的事情，Storm让事情变得非常简单。

批处理层+实时层、CAP定理和人为错误容忍性

貌似又回到一开始提出的问题上去了，访问实时数据需要使用NoSQL数据库和增量算法。这就说明回到了版本化数据、矢量时钟和读取修复这些复杂问题中来。但这是有本质区别的。由于实时层只处理最近几小时的数据，所有实时层的计算都会被最终批处理层重新计算。所以如果犯了什么错误或者实时层出了问题，最终都会被批处理层更正过来，所有复杂的问题都是暂时的。

这并不意味着不需要关心实时层的读取修复和最终一致性，你仍然需要实时层尽可能的一致。但当犯了一个错误时，不会永久性地破坏数据。这便移除了许多你所需要面对的复杂问题。

在批处理层仅需要考虑数据和数据上的查询函数，批处理层因此很好掌控。在实时层，需要使用增量算法和复杂的NoSQL数据库。把所有的复杂问题独立到实时层中，对系统的鲁棒性、可靠性做出了重大贡献。

同样的，实时层并没有影响系统的人为错误容忍性，这个数据不可变和只追加的批处理系统，仍然是整个系统的核心，所以所有的都可以像上面说的一样被纠正过来。

我有一个类似的系统：Hadoop和ElephantDB组成批处理系统，Storm和Cassandra组成实时系统。由于缺乏监控，某天当我起床的时候发现Cassandra运行满负荷了，使得所有的数据请求都超时。这使得Storm计算失败，一些数据又重新回到了等待队列中，这个数据就一次次重复请求。

如果我没有批处理层，那么我就需要扩展和恢复Cassandra，这个很不容易。更糟的是，因为请求不断的重复，无法得到正确的数据。

幸运的是，所有的复杂问题都被隔离到实时层中去了，我清空了所有的后台请求队列，把它们打到了批处理层上，同时重启了Cassandra集群，过了几个小时之后所有数据都恢复正常了。没有错误数据，请求中也没有不准确的地方。

垃圾回收

上面描述的所有东西都是建立在一个不可变的、不断增长的数据集上的。如果数据集已经很大，使得不可能用水平扩展储存所有时间的所有数据，该如何处理呢？这是不是就推翻了我说的一切呢？是不是需要回到可变数据的系统上呢？

不。我们可以很容易地用“垃圾回收”对基本模型进行扩展来解决上面的问题。垃圾回收是一个在主数据集上的简单函数，返回的是一个过滤版本的主数据集。垃圾回收掉了旧数据，可以选择任意的垃圾回收策略。可以在易变的系统中只保留数据最新的一个值或者保留每个数据的历史。比如，如果要处理位置数据，可以保留每人每年的一个地点。可变性是一个不是很灵活的垃圾回收形式（它跟CAP定理交互得也很糟糕）。

垃圾回收可以被实现成批处理的一个任务，隔段时间运行一下。由于它是作为离线批处理任务执行的，所以不影响我们与CAP定理的交互。

总结

让可扩展的数据系统复杂的原因不是CAP系统，而是数据增量算法和数据的可变状态。最近由于分布式数据库的兴起导致了复杂度越来越不可控。前面讲过，我将挑战对传统数据系统构建方法的假设。我把CRUD变成了CR，把持久化层分成了批处理和实时两个层，并且得到对人为错误容忍的能力。我花费了多年来之不易的经验打破我对传统数据库的假设，并得到了这些结论。

批处理/实时架构有许多有趣的能力我并没有提到，下面我总结了一些。

- 算法的灵活性。随着数据量的增长，一些算法会越来越难计算。比如计算标识符的数量，当标识符集合越来越大时，将会越来越难计算。批处理/实时分离系统给了你在批处理系统上使用精确算法和在实时系统上使用近似算法的灵活性。批处理系统计算结果会最终覆盖实时系统的计算结果，所以最终近似值会被修正，而你的系统拥有了“最终精确性”。
- 数据结构迁移变得很容易。数据结构迁移的难题将一去不复返。由于批量计算是系统的核心，很容易在整个系统上运行一个函数，所以很容易更改你数据的结构或者视图。
- 简单的Ad-Hoc网络。由于批处理系统的任意性，使得你可以在数据上进行任意查询。由于所有的数据在一个点上都可以获取，所以Ad-Hoc网络变得简单而且方便。
- 自我检查。由于数据是不可变的，数据集就可以自我检查。数据集记录了它的数据历史，对于人为错误容忍性和数据分析很有用。

我并没有说我已经“解决”了数据量过大的问题，但我已经为解决大数据问题制订了一个框架。批处理/实时架构可以应用到任何一个数据系统中去，“授人以鱼，不如授人以渔”，我已经告诉你了如何去构建这样的系统。

为了提高系统整体能力来解决大数据的问题，我们还有许多工作需要做。

- 扩展数据模型，支持批量写和随机读。不是每一个应用程序都支持key/value的数据库，这也是我们团队对扩展ElephantDB，使得可以支持搜索、文档数据库、区间查询感兴趣的原因。
- 更好的批处理原语。Hadoop并不是批处理的最终形态，好多批处理计算Hadoop效率不高。Spark是一个有意思的扩展MapReduce的项目。
- 提升后的读写NoSQL数据库。这里不同类型数据的数据库还有很大的提升空间，随着这些数据库的成熟，它们将收获很多。
- 高层级的抽象。未来工作中最有意思的就是对批处理模块和实时处理模块的高层次抽象，在批处理和实时架构下你没有理由不拥有一个简单的、描述性的、鲁棒性好的语言。

许多人需要一个可扩展的关系型数据库，本文就是想让你知道完全不需要那个。大数据量和NoSQL运动使数据管理比RDBMS更加复杂。那仅仅是因为我对大数据的处理采用了跟RDBMS同样的方法：把数据和视图混为一谈，并且依赖增量算法。大数据量需要采用完全不同的方式构建数据系统。通过存储持续增长的不可变数据，并且系统核心采用预计算，大数据系统就可以变得比关系型数据库更易掌控，并且可扩展性很强。

（感谢Nathan Marz先生的授权，原文名为*How to beat the CAP theorem*。同时感谢方建对本文翻译做出的贡献。）

责任编辑：董世晓（dongsx@csdn.net）

软硬兼施，直指海量数据

解析内存计算技术HANA

文 / 董世晓

海量数据时代，人们已不再为信息匮乏而苦恼，信息也变得不再昂贵，而如何从海量信息中获取价值却变得越发昂贵。

企业IT需求的现状

当前，各行业对IT需求的趋势越来越呈现出以下特征：

- 期望实时决策；
- 希望需要在面临风险的情况下作出决策；
- 在适当的时间及时地提供产品；
- 用户希望更好的体验；
- 需要即时及正确地应对任何问题。

面对这些特征，现实情况却是，企业往往受限于高效管理大量数据、面对问题实时获取答案、正确而完整的信息这三大看起来是无法同时满足的需求。

之所以出现这些问题，原因无非以下几点：

- 磁盘I/O成为瓶颈，导致延迟及CPU的低效利用；
- 关系型数据库为处理交易数据而设计；
- 数据存储的方式无法高效满足分析需求；
- 80%BI查询使用不到20%的获取的行数据。

HANA出世

正是基于以上分析，以及内存价格下降、大型内存模块已经商用、CPU主频不再变快、CPU内核数量不断增加等应用背景，内存计算技术得以提出。目前，内存计算技术已经实现了以下功能：支持在服务器的主内存中处理超大量的实时数据，从分析和交易中提供即时的分析结果。

作为一种海量数据实时分析工具，HANA能在不改变现有IT信息系统架构的前提下，实时分析来自几乎任何数据源的海量数据，提高计算速度，从而减少企业数据量，降低总体拥有成本、开发工作量/时间、事件记录和报告呈现的时间延迟、企业数据管理等，大幅提升用户应对市场变化的能力。HANA架构如图1所示。



图1 HANA架构图

长久以来，一直流行的误区是OLTP是写入导向的，OLAP是读取导向的。可以说OLTP与OLAP并没有那么大的区别。目前的应用架构为不同的应用创建专用的数据仓库，冗余数据降低了实时性能，也增加了系统的复杂性。

HANA这种新的内存计算技术，使得：

- 列存储成为数据类型的主要存储方式；
- OLTP和OLAP使用相同规范的数据结构；
- 外部数据源数据通过复制或ETL进入系统；
- 操作是海量并行的，随着核数的增加，可容许操作数线性增加；
- 所有聚集是按需和实时的，没有汇总和数据立方；

■ 精细的图形化SQL开发环境、OLAP计算、规划方法、模式搜索算法、文本和业务函数库；

■ 对于结构化和非结构化数据的SQL操作；

■ 磁盘存储只用于备份和恢复。

用事实说话

实践是检验真理的唯一标准，再完美的理论如果不能付诸应用都将是空谈。HANA也于近日在中国得以实施——农夫山泉成为全球第三家成功应用HANA的企业。

农夫山泉业务涉及生产、销售、计划、调度、物流、市场营销等多个方面，对企业的综合管理和整体运营能力提出了很高的要求。同时，如何在瞬息万变的经济环境中根据实时的数据分析作出准备、快速的决策，对企业发展和竞争力都至关重要。因此，农夫山泉要寻找一个解决方案，以应对海量数据的挑战，实现高效的逻辑运算、实时的数据分析和快速的数据展现。

据悉，经过对比测试，同一个脚本在SAP HANA Studio中运行要比某平台的返回结果快200~300倍，而且这种查询性能的提高对于150多张报表的表现都高度一致；在某平台中要24小时才能执行完的存储过程和功能，HANA只需要37秒；在对账流程上，HANA帮助农夫山泉整整缩短了1天的时间。HANA也带来了额外的益处——通过将来自SAP系统和其他异构数据源的数据实时同步到HANA，使农夫山泉摆脱了传统的IT运维和数据滞后问题，整个生产环境正呈现出更顺畅的运营、更准确的数据和更高效的性能。

旧的IT系统架构

农夫山泉原先的架构如图2所示。

这个阶段，OLAP遇到了数据展现速度越来越慢如下挑战、数据运算难以忍受等问题。因此，农夫山泉希望通过HANA实现以下期望：

■ 快速：利用HANA内存计算技术能够将基于详细数据的报表展现时间提升到10秒以内；

■ 实时：将凭证流实时更新到OLAP；

■ 高性能：逻辑计算能力提升。

实施HANA后的架构

农夫山泉实施HANA经历了两个阶段，第一阶段是将数据集市ETL到HANA，架构如图3所示。第二阶段是采用HANA的同步数据传输技术实时将SAP和非SAP系统复制到HANA，如图4所示。

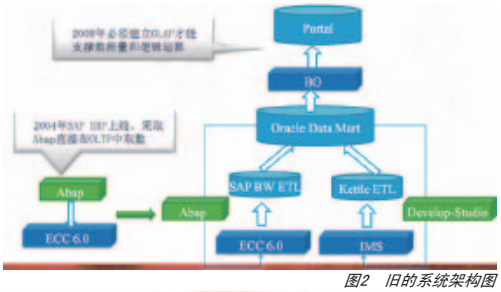


图2 旧的系统架构图

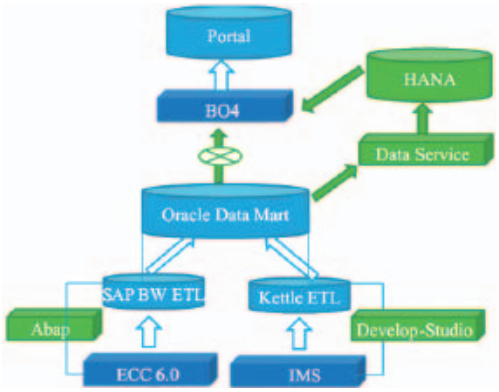


图3 将数据集市ETL到HANA

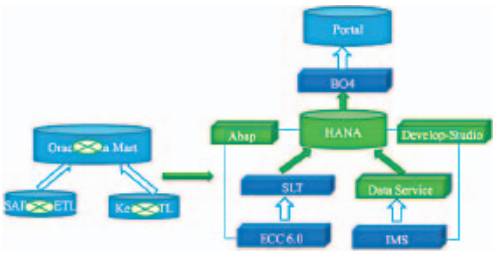


图4 将SAP和非SAP系统复制到HANA

经过上面的改进，农夫山泉实现了：

■ 同样的报表在BO4 for HANA与其他相比速度更快；

■ 同样逻辑的SQLScript在HANA与其他相比运算速度更快；

■ HANA可以作为传统数据库被外部系统调用或作为SOA接口部署；

■ SLT的精彩实时同步。

结束语

HANA作为一种开放性的数据库解决方案，在实时海量数据处理方面体现出了自己的特点和优势，也初步得到了市场的认可。而它能够进一步得到普及和推广，值得期待。

漫谈基于HTTP协议的SOA架构

文 / 陈康贤

基于HTTP协议的SOA架构近来被广泛使用，无论是新浪微博、腾讯微博的开放平台，还是淘宝开放平台，无一不是基于HTTP协议架构进行设计的，本文将详细论述基于HTTP的SOA架构以及亟待解决的一些问题。

在互联网时代，日益复杂的应用系统、不断变换的商业环境带来的无穷无尽的业务需求,使得快速实现满足业务要求的应用系统遭遇到严重挑战，各个项目之间容易形成孤岛，常常导致重复造轮子，因此构建企业的SOA（Service-Oriented Architecture）体系结构，对于企业来说越来越重要，也越来越成为支撑企业业务发展的重要因素。而基于HTTP协议的SOA体系结构，具有如下优点。

- 支持异构平台调用：由于采用的是HTTP协议，服务中心实现对调用方透明，只需按一定格式返回数据，就可以支持多种语言平台的调用,如Java、PHP、Python、Ruby和.NET等。
- 扩展方便：如果系统负载大或者需要进行业务扩展，那么横向和纵向的扩展均非常容易。
- 无状态：HTTP协议是无状态协议，意味着服务中心可以很快做出响应，服务器处理完客户请求，并收到客户的应答后，即断开连接，这样可以降低系统负荷。
- 灵活：对于C/S结构和B/S结构的调用都能很好支持，并且HTTP允许传输任意类型的数据对象，传输的类型由Content-Type加以标记。

因此，该体系结构越来越成为企业的首选。

HTTP服务中心架构

HTTP服务中心架构如图1所示。

服务调用方为了标识身份，需要提供appKey和sign，调用方需要注册appKey和向服务中心申请appSecret，然后由appSecret经过一定的算法生成sign，请求才被认为是合法的。

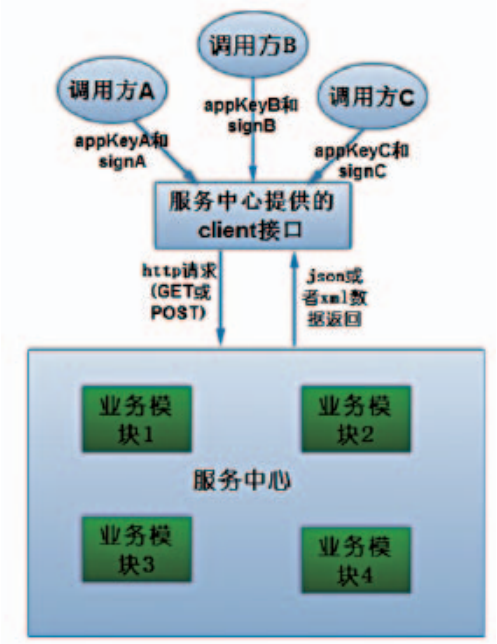


图1 服务中心架构图

服务中心对调用者的appKey和appSecret进行统一管理，并且验证用户请求是否为经过授权的合法请求。

Client由服务中心提供，目的是给调用方提供一个友好标准的接口，避免调用方直接拼装URL参数和解析JSON或者XML数据，造成格式不统一，简化了调用，并且Client可以通过多种语言来实现(如Java、PHP、C#等),以方便不同平台调用。当然，调用方完全可以自己来完成这一工作，但需要按照服务中心提供数据格式来

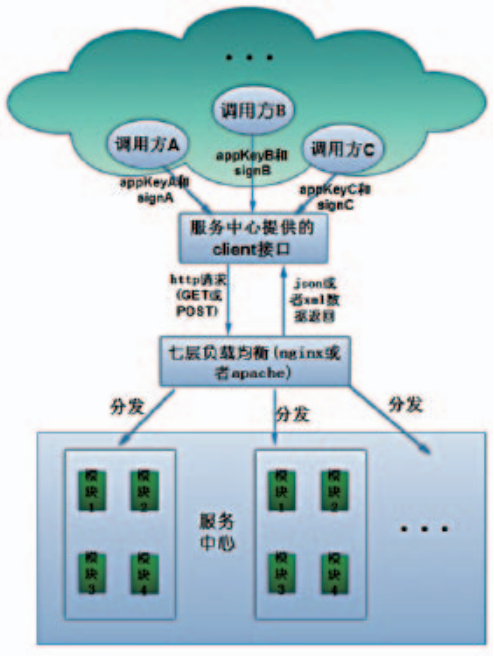


图2 服务中心横向扩展架构图

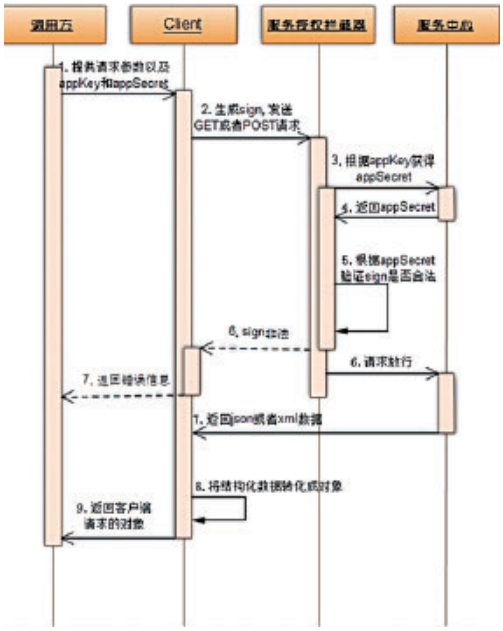


图3 授权验证过程

进行，否则调用无法完成。

随着企业的发展，很可能会面临服务中心负载过高以及新业务需求不断的到来等问题，这时就需要对服务中心进行扩展。扩展分为横向扩展和纵向扩展，图2是横向扩展图，纵向扩展类似，便不再赘述。

可能遇到的问题

在服务中心的构建过程中，可能会碰到下面一些问题。

服务的安全性

服务中心是整个企业数据中心的入口，如何校验请求的合法性便成为首当其冲的问题，考虑到HTTP协议的特有属性，请求的合法性验证必须包括两个方面，即调用方身份的验证以及参数是否被篡改，图3是我的一个解决方案，供大家参考。

服务调用方先必须申请appKey和appSecret，并将appKey和appSecret与参数一并传递给Client接口，由Client接口统一构造HTTP请求，传递给服务中心。而服务中心在做出响应之前，请求必须先通过服务授权拦截器的验证，只有验证通过的请求，才能够被放行。

请求可控

由于服务调用者众多，调用行为难以预测，作为服务提供方，服务中心必须稳定可靠的提供服务，不能受某一调用者影响而对其他调用产生干扰。因此，请求必须是可控的，以免请求超过服务中心的设计上限。大多数网站使用Nginx来做七层负载均衡，而Nginx从0.7版开始提供2个限制用户连接的模块NginxHttpLimitZone Module和NginxHttpLimitReqModule，可以用来限制用户请求的频率和规模：

```
limit_zone   zone_session_state   $binary_
remote_addr  10m;
limit_req_zone   $binary_remote_addr
zone=req_session_state:10m  rate=1r/s;
location /
{
    limit_conn  zone_session_state 1;
    limit_req   zone=req_session_state
burst=5 nodelay;
    …   …
}
```

当然，大家也可以举一反三，把这种思路应用到Apache和Lighttpd等常用的反向代理服务器当中。

标准统一

作为服务提供方，提供统一的调用形式和数据格式是一件极其重要的事情，如果各接口标准各异、参数混乱，调用者便无所适从，

所以标准的制订应包括请求的参数及URL的格式、返回数据的格式，以及调用的相关接口。

URL一般可以按照这样的形式定义，如http://api.domain.com/service/rest?sign=ABCDEFG&timestamp=2011-10-10 22:28:10&app_key=12345&method=servicecenter.item.get&format=json&id=100，其中，sign是该次请求调用方生成的签名信息，timestamp为用户请求的时间戳，app_key是调用方的标识，method对应的是请求的服务中心具体的业务方法，format表示返回的数据为JSON格式，而后面可以带上跟业务有关的参数，如id等。

至于返回数据的格式，一般均采用JSON或者XML，这样可以方便异构平台之间进行交互。

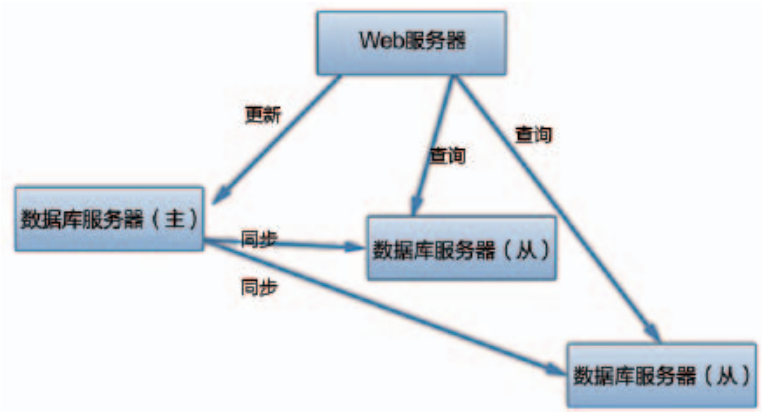


图4 数据库主从配置

数据库性能

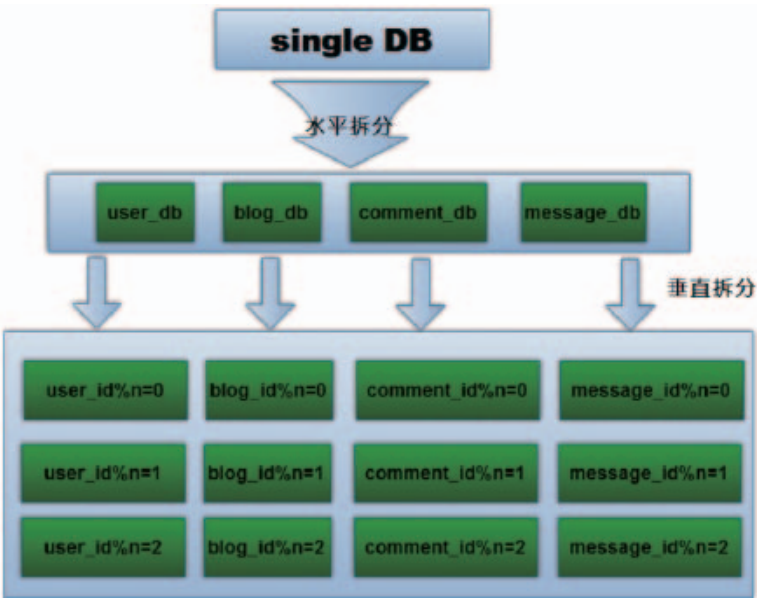


图5 数据库水平和垂直拆分

随着业务的拓展，服务中心的压力不断增加，为了保证服务中心在任何时刻都能够稳定地提供服务，必须提前进行容量规划，在性能还未达到极限时，对系统进行扩容。

最开始，压力较小，可以采用主从同步、读写分离方式。一般而言，大多数站点读操作比写操作更加密集，很大一部分开销都消耗在查询操作上面，而通过读写分离，我们可以将大量的读操作剥离出来，通过水平扩展，将压力转移到更多的从服务器上。

随着数据库表的记录的增长以及访问压力越来越大，原有的架构达到极限，这时我们不得不进行改变，以MySQL为例，单表的记录数超过100万时性能便会显著下降，因此我们必须对表进行拆分。同时，当数据库主服务器无法承受写操作压力时，不管你如何扩展从服务器，此时都已经没有意义了，因此我们必须换一种思路，对数据库进行水平和垂直拆分。当然，拆分方法必须跟实际应用场景关联，我提供的拆分方式仅供参考，如图5所示。

当然，切分也可能会带来种种问题，最明显的就是数据库的架构以及应用程序的架构将愈加复杂，维护起来愈加不便，而且拆分会导致系统各部分耦合度增加，对后面的数据迁移造成很大影响。

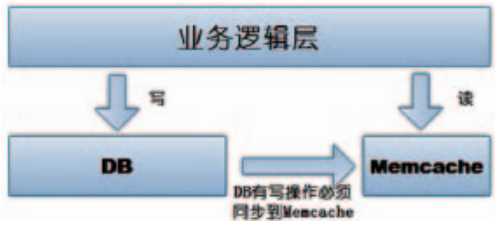


图6 缓存

缓存

为了减轻DB的压力，企业一般会在DB前面加一层缓存，将经常访问的数据置于缓存当中，把磁盘的机械动作转换为对内存的直接访问，这其实是一个效费比很高的操作，既简单又能显著提高性能，如图6所示。

常用的数据或者小表，直接将其缓存在Memcache当中，业务逻辑层从Memcache读取，当需要对DB数据进行写操作时，同步对

Memcache进行更新，否则将会造成Memcache中数据与DB中数据的不一致。

统一Session存储

对于服务中心来说，支撑其业务的远远不止一台服务器，而是一个分布式集群，在不同服务器之间跳转，如何保持Session的同步呢？传统网站可以通过将一部分数据存储在Cookie中，来规避分布式环境下Session的操作，但服务中心面对的是调用者，而不是终端用户，因此必须采取其他办法。

对于系统可靠性要求较高的用户，一种解决方案是可以将Session持久化到DB中，这样可以保证宕机时Session不易丢失，但缺点也是显而易见的：系统的性能将受到影响。另一种解决方案便是将Session统一保存在Memcache集群上，这样可以保证一定的访问性能，但如果Memcache重启，里面存储的Session也就丢失了。究竟采用哪种方案，需视具体情况而定，不能从一而论。

我以tomcat为例，对以上两种方案均一一做了实现，并提供源码供大家参考，详情见我的博客。

系统分层

大型系统如果不采用分层结构，团队之间协作将非常困难，这是一个不可回避的问题。这里所谓的分层，并不涉及物理设备的分层，而是软件结构的分层。一般的分层会是这样：模型层、数据访问层、基础接口层、服务层、应用层。各层自顶向下调用，而平行层之间不进行相互调用。

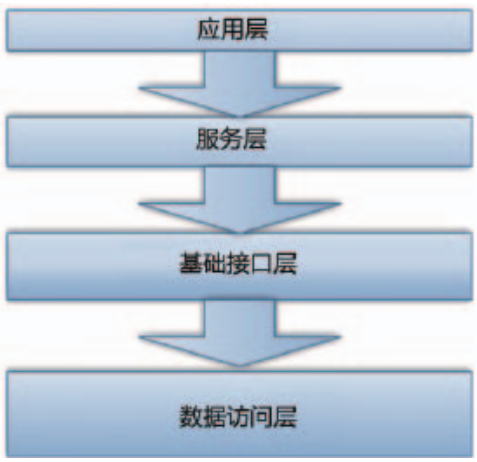


图7 系统分层

- 数据访问层，就是应用程序中对数据库进行访问的一层，提供数据库的基本增删改查操作。
- 基础接口层建立在数据访问层之上，这一层主要提供对数据访问层的封装，以及缓存的操作、搜索引擎的调用和消息队列的操作。
- 服务层是利用基础接口层提供的一些接口，对应用层提供服务，这一层将包含一定的业务逻辑。
- 应用层依托服务层和基础接口层，来完成一系列功能。

小结

本文对HTTP服务中心的架构给出简要介绍，并讨论了架构HTTP服务中心将面临的七大问题。HTTP服务扩展方便，使用灵活，适合异构平台之间的相互调用，相比较其他解决方案(如WebService、RMI、Socket等)，更加容易实现，系统间的耦合度更低，开发起来更加快速。

但由于HTTP协议本身的一些特性，也可能会存在一些不足，如网络延迟、安全等，因此对于实时性要求很高的交互（如网络游戏）以及安全等级很高的应用（如银行内部交互）来说，可能不大适用。

总之，对于企业级SOA架构来说，基于HTTP协议的服务中心架构无疑是一种既实用又简捷的方式。



陈康贤
阿里云无线运营部开发工程师，主要研究方向为J2EE，目前关注于Java高性能程序设计及JVM调优，ITeye博客地址：http://chenkangxian.iteye.com。

责任编辑：董世晓（dongsx@csdn.net）

寻找机遇 创造未来 CSDN 热门职位全新推荐

杭州优朋电子商务有限公司

杭州优朋电子商务有限公司位于风景秀丽的美丽之洲——杭州余杭，是一家高新技术企业，专业从事计算机软件技术研发、信息技术服务及旅游电子商务网络运营。公司独立开发运营的旅游电子商务平台，为国内提供一站式服务的个性化旅游服务平台。

因业务发展需要，急需以下人才加盟，让我们一起来创造美好的明天！

- .NET高级软件开发工程师
- .NET中级软件开发工程师
- Web前端开发工程师
- 网站运营总监

简历投递：zhuqr@uniboncn.com 联系QQ：5462240

欧特克软件（中国）有限公司

你一展身手的舞台，打造美好未来

Autodesk

加入我们的队伍，以一种全新的方式实现设计理念。无论是设计可持续性建筑还是开发混合动力汽车，Autodesk软件都可助你一臂之力。作为全球前端的二维与三维设计、工程和娱乐软件提供商，我们致力于帮助打造出更美好的未来。

让我们共同创造未来!

更多职位信息请浏览网站：www.autodesk.com/careers
或直接将简历发送至：acrd.hiring@autodesk.com

方正宽带网络服务股份有限公司

FOUNDER 方正
正 在 你 身 边

公司诚聘：

- 高级研发工程师 (Linux C方向)
- 高级研发工程师(Java方向)
- 高级研发工程师(PHP方向)
- 软件测试工程师

任职要求：

- 本科及以上学历，计算机或网络相关专业
- 熟悉相应的开发语言，有相关开发经验者优先
- 良好的团队合作精神，较强的创新精神

发送简历请以“姓名+职位+CSDN”为标题
邮箱：hr01@founderbn.com
公司网址：http://www.founderbn.com

聚胜万合信息技术（上海）有限公司

聚胜万合MediaV依托强大的互联网广告平台，为广告主、网站主和战略合作伙伴提供专业的互联网营销服务。现热招云计算工程师、PHP工程师、数据处理应用工程师、产品经理等职位，邀您共谋中国数字营销大变局。如果您想在互联网广告领域一展鸿图，如果您拥有足够的经验和才能，MediaV期待您的加入。

MediaV

公司网址：http://www.mediav.com

杭州拓乐网络科技有限公司

Tolez 拓乐网络

诚聘高级网游程序员若干：

- 逻辑程序员
- 客户端程序员
- 服务器程序员
- 网络工程师和计算机管理员

更多岗位信息请访问公司网站： www.tolez.com

简历投递：hr@tolez.com 联系人：徐小姐

厦门梦加网络科技有限公司

客户端程序员 （重点职位） 若干

至少一年相关工作经验，熟悉C/S架构，拥有良好的学习能力和独立解决问题的能力。根据游戏策划和主程序的要求，负责游戏客户端编程和维护工作。

MECHANIST GAMES

招聘要求：

1. 计算机相关专业，熟练掌握Java或C#编程语言，热爱游戏编程；
2. 使用过Unity3D引擎开发过游戏者优先；
3. 公司统一使用英文版本的技术软件，拥有大量外籍员工，英语口语较好者优先。

公司地址：厦门市思明区厦禾路666号海翼大厦A座11层
公司主页：www.mengjiagames.com
公司电话：0592-2661183骆小姐 简历投递：hr@mechanist.co

完美世界（北京）网络技术有限公司

完美世界诚聘各路英才，欢迎与我们一起开启一个完美世界

招聘职位：

- 游戏引擎开发工程师
- Java开发工程师
- Flash (AS3)开发工程师
- PHP开发工程师

详细JD请登录http://hr.wanmei.com/

简历投递邮箱：duzhipeng@pwrd.com
QQ咨询：2393130592　**欢迎大家投递简历**

天津港保税区科技发展局

东软（天津）、展讯（天津）高薪诚聘IT英才

招聘职位：高级Java工程师，高级C++工程师，DSP工程师，芯片驱动工程师，应用软件工程师，驱动软件工程师，算法工程师

工作地点：

天津空港经济区（www.tjftz.gov.cn）

简历投递：resume@adm.tjftz.gov.cn

北京法国电信研发中心

Software engineer in the web service system development

Position requirements:

- Master degree in telecommunication, Compute Science OR equivalent
- Experience in J2EE, SOA, Spring, Hibernate, Struts, XML, REST, HTML5 development
- Experience in cloud computing development, such as SaaS, PaaS, etc.
- Strong programming skills, at least 3+ years experience in Java software development
- Familiar with the software development methodology
- Familiar with networking technology
- Good English communication skill
- Good team working spirit and communication skills, and hard working as well

Contact：hr@orange-ftgroup.com.cn

广州瀚信通信科技股份有限公司

招聘职位：

- C/C++ 初级开发工程师
- C/C++ 中级开发工程师
- C/C++ 高级开发工程师
- C/C++ 系统架构师
- C# .NET初级开发工程师
- C# .NET中级开发工程师
- C# .NET高级开发工程师
- C# .NET系统架构师
- 研发项目经理（C/C++或.NET、Delphi等均可）
- 软件测试工程师
- 交互设计师
- 交互设计师助理

详细信息请登录CSDN网站查询！请以“职位+姓名+CSDN”作为邮件标题，将简历发送至：yfhr@hantele.com

无锡迈达博尔信息咨询有限公司

无锡国家软件园，占地800亩，建成55万平方米，在建37万平方米，入园企业超400家。
网址http://wnd.iparkhr.cn/

招聘职位：

- 无锡国家软件园——对日软件工程师
- 无锡国家软件园——Java开发工程师
- Android开发工程师
- 测试工程师

简历投递：iparkhr@126.co

方正国际软件有限公司

方正国际医疗卫生业务是中国医疗信息系统解决方案的提供商，经过16年的发展，已成为成熟的数字化医院及区域医疗卫生网解决方案的提供商和服务商。

为适应业务的蓬勃发展，现在急聘

Java中高级开发人员　Unix/C高级开发

投递邮箱：bfi_zhaopin@founder.com
公司主页：www.founderinternational.com
公司地址：北京市海淀区北四环西路52号方正国际大厦5层

创建可维护的自动化验收测试

作者 / Dale. H.Emery 译者 / 张小明，Holly

本文原稿首次发表在2009年的敏捷开发实践大会上，彼时是作为敏捷测试研讨分会系列的一部分，此番再刊作者做了一些更新。

测试即开发

这里的测试指的是自动化测试，从软件的本质上看，测试的自动化乃是测试方面的软件开发，万变不离其宗，这也就意味着那些凡是属于软件开发的定律或者原则也同样适用于测试自动化。对于没有写过代码或者代码经验较少的人来说，或许这其中的道理不能一眼就瞧得出来。

通常情况下软件开发的很大一部分开销是维护——修修补补，更新不断。软件的可维护性强，则开发成本低，同理，测试的自动化开发成功与否也很大程度上体现在它的可维护性成本大小上。我接触过的很多试图尝试引进自动化测试的机构没几个月就决定放弃自动化测试，问之弃因，你会发现大多是因为自动化脚本过于不稳定以及随之衍生的难维护性。举个例子，界面上重命名一个按钮会导致大批的测试用例失败，而与此同时花费在调通和更新这些用例身上的时间成本又太高。

有些团队或机构在自动化测试上取得了成功，难道他们的自动化就可以避免掉这样的维护费用问题？当然不可能。而成功与失败的团队之间一个重要的区别就是：在对待测试开发的维护问题上，失败者往往是被昂贵的维护费用吓住而放弃自动化计划，而成功者则是从一开始就做足了应对措施。那些在自动化取得成功的团队懂得测试即开发这一道理，明白测试开发一旦开始，维护在所难免，所以他们会深思熟虑，想方设法降低维护成本。

软件需求的变更和系统实现的变更会影响测试，需要测试做出相应的调整，这二者任意一种变更都可能导致一系列的自动化测试失败。如果一些自动化测试不能同步新的软件变更和产品新特性，那么它们将会被淘汰，其测试结果也不会得到运用。而要使其回归正常，我们必须不断调整测试以配合需求和系统实现的变更。维护的成本开始显山露水。

因此如果需求和实现的变化是必然的，那么降低自动化测试维护成本的方法只有一个，即编写适应性强的测试脚本。

暴露太多无关紧要的细节或者重复这两大关键因素使得修改代码的困难大大增加，无数惨痛的经验教训让软件开发人员对此深有体会，对于那些正在从事和将要从事的自动化测试开发者们，也肯定不想重蹈覆辙。

验收测试和系统的任务

验收测试用来检测一个系统是否正确履行了某一特定任务。也就意味着，验收测试的核心是关注它所要验证的功能点是否正确，而不考虑用了何种技术、何种方法去测试。

现在假定我们要测某个系统的创建账号这个特性，系统通过传递给Create命令用户名和密码来创建新账号。创建账号特性的功能之一是验证密码的有效性。一个合法的密码长度必须介于6~16字符之间且至少包含一个字母、

一个数字以及一个标点符号。如果用户提交的密码合法，Create命令创建成功并报告Account Created；反之，Create命令不会执行创建过程，同时报告Invalid Password。这就是功能职责的本质。无论软件系统以何种技术实现，Web应用也好，GUI桌面应用也罢或者是命令行执行的程序，也不管会不会有人像德州电锯杀人狂里的休维特一样，挥舞疯狂的电锯恐吓要锯断那些输错密码童鞋的指头，总之，系统需执行此项职责（密码检查是系统必须实现的职责）。

无关紧要的细节

列表1展示的是一段不良自动化测试用例脚本，该测试用例用来检测Create命令的密码有效性检查这一职责。

表1　不良自动化测试用例脚本

Listing 1: A poorly written acceptance test

```
** Test Cases **
The create command validates passwords
    ${status}=  Run  ruby app/cli.rb create fred 1234!@$^
    Should Be Equal  ${status}  Invalid Password
    ${status}=  Run  ruby app/cli.rb create fred abcd!@$^
    Should Be Equal  ${status}  Invalid Password
    ${status}=  Run  ruby app/cli.rb create fred abcd1234
    Should Be Equal  ${status}  Invalid Password
    ${status}=  Run  ruby app/cli.rb create fred !2c45
    Should Be Equal  ${status}  Invalid Password
    ${status}=  Run  ruby app/cli.rb create fred !2c456
    Should Be Equal  ${status}  Account Created
    ${status}=  Run  ruby app/cli.rb create fred !2c4567890123456
    Should Be Equal  ${status}  Account Created
    ${status}=  Run  ruby app/cli.rb create fred !2c45678901234567
    Should Be Equal  ${status}  Invalid Password
```

这段测试脚本问题很多，一眼望去，最明显的是可读性很差，我们看到第二行The create command validates passwords，这是测试的标题，表明该测试的测试点和职责，但往下读时，我们会发现，里面充斥了太多累赘的单词和烦人的诸如“{$@^”这样的符号，让人不知其所言。

仔细看一下，我们可以挑出来几个密码，比如1234!@$^!紧接着再加把劲，啊哈，我们会发现一些密码会导致status值为Invalid Password，而另一些会使得status值为Account Created。从另一方面看，我们可能也会注意不到上述内容因为该测试脚本在密码和状态status之间夹杂了太多的实现细节，或曰：测试噪声。试想，这些特殊的符号$、@、^以及单词Run、Ruby、fred究竟和密码及其有效性有什么关系!对于自动化测试的脚本来说，从用例的可读性和可维护性角度看这些都是无关紧要的过程实现细节。

过多无关紧要的细节是怎样毁掉可维护性的？假设我们的系统安全分析师指出六位长度的密码本身不安全。于是为了增强安全性，我们将密码长度下限由六改成十，这是一个典型的需求变更，请思考，这时候列表1的测试脚本哪里需要修改？怎样改？答案恐怕没那么简单。

让我们再来考虑一个更有挑战性的需求变更。假如我们想让系统管理员能够为任一种情况设定具体的密码长度最长与最短值。这时候该怎么修改刚才的测试脚本？答案还是无法一眼看出。恐怕没那么简单。

这其中的原因“恐怕”在于测试脚本没有清晰表达它所要测试的功能职责。当看不出一段测试用例脚本的本质，通常意味着需求变更之时测试人员会需要花数倍的代价来修改原测试脚本。

因此，为方便识别本质就要隐藏非必要细节的，以使自动化脚本用户更容易看到测试的本质，在上面创建用户的例子中，大多数非必要的细节是如何调用Create命令。该系统是基于Ruby的命令行程序，现在让我们再次返回列表1的测式脚本里解读一番，黑色字体加深的第一行告诉自动化测试框架Robot启动Ruby解释器，加载被测程序文件app/cli.rb，并调用Create命令，参数值为用户名fred以及密码1234!@$^，最后命令返回的结果存在变量${status}中，呵，数数不必要的实现，细节至少有5~6个之多!

再来看字体加深的第二行，实现命令返回值和期望值Invalid Password的比较，虽然看起来比刚才那一行较容易理解，但措辞笨重，并且过分的语法细节容易分散人们的注意力。

通过Robot自动化框架我们可以把实现细节提炼成关键字（Keyword），使之以类似子函数的形式为测试用例脚本调用，一个完整的自动化测试用例便可由数个关键字组合而成。

现在我们演示如何使用关键字来隐藏不必要的实现细节。一个可行的方法就是问自己这样一个问题：假设自己对被测系统实现一无所

知，该如何写出自动化测试脚本的第一步？是的，即使对实现一无所知也无大碍，我们只需知道我们要测试创建用户这个产品特性——被测系统显然要提供的功能。继而我们知道创建用户即是被测系统的必要职责，而且从系统需求分析可知创建用户需要提供用户名和密码。

基于以上所述，可能这样修改测试：

```
Create Account fred 1234@!$^
```

当然修改后的脚本可能还有其他一些问题，我会在稍后部分接着讨论。

再看看加深的第二行，验证创建用户命令返回的结果是否为Invalid Password,顺着上面修改的思路则可以变成：Status Should Be Invalid Password。

两行合并，看一看整体效果：

```
Create Account fred 1234@!$^
Status Should Be Invalid Password
```

原来的一步经过一次提炼现在看起来简洁多了，可读性也变强了，我们很容易发现这两行的逻辑上的联系：系统必须告知输入的这一密码是无效的。

现在还无法运行新的脚本，因为测试框架Robot找不到关键字Create Account和Status Should Be的定义，两个关键字的Ruby实现代码如表2。

表2 两个关键字的Ruby实现代码

Listing 2: Keywords to create an account and check the status

```
** Keywords **
Create Account ${user_name} ${password}
    ${status}=  Run  ruby app/cli.rb create ${user_name} ${password}
    Set Test Variable   ${status}

Status Should Be ${required_status}
    Should Be Equal  ${status}  ${required_status}
```

字体加深的代码行创建了一个名为Create Account的关键字，需要两个参数user_name和password。关键字函数的主体只有两行代码，第一行加载被测程序并调用Create命令，第二行保存返回值，对比之前的原测试脚本，我们发现主体第一行代码和表1的加深第一行实现了同样的功能，非必要细节即隐藏于此。

你或许已经发现了关键字的实现代码里引入了更多的语法和特殊字符。这不用多虑，通过把细节抽象成关键字， 我们的测试脚本看起来整洁多了，可读性大大增强，现在我把更新过的测试脚本贴上如表3所示，更直观展示了修改后的效果。

表3 更新过的测试脚本

Listing 3: The test rewritten to remove incidental details

```
** Test Cases **
The create command validates passwords
    Create Account fred 1234!@$^
    Status Should Be Invalid Password
    Create Account fred abcd!@$^
    Status Should Be Invalid Password
    Create Account fred abcd1234
    Status Should Be Invalid Password
    Create Account fred !2c45
    Status Should Be Invalid Password
    Create Account fred !2c456
    Status Should Be Account Created
    Create Account fred !2c4567890123456
    Status Should Be Account Created
    Create Account fred !2c45678901234567
    Status Should Be Invalid Password
```

虽然某些程度上增加了关键字这一部分的代码，但获得了整个测试用例脚本的干净清爽，这一做法是值得尝试的。

重复

上面我们已经学会通过提炼可复用的关键字改进了测试脚本，但还存在其他问题。一个问题是我们之前提到的，即每隔一个测试步骤就包含用户名fred。另一个更大的问题是重复，从表3修改过的测试来看，每一对关键字（Create Account和Status Should Be）组成一步验证测试，每一步都要提交一个不同的密码并与比较系统返回的状态值和期望值，我们看到，除了输入的密码和期望状态值不同之外，其他部分基本都是一样的。

重复的代码将毁掉可维护性。现在假设我们的用户交互分析师指出产品系统的其他部分并不要求用户创建账号（Create）而是要求他们注册（Register），这就在同一个系统里出现了用户交互接口和使用术语的不一致，而用户交互分析师坚持整个系统应该杜绝这种不一致性，于是我们决定把Create命令变成Register。

这样一来，对测试会产生的影响有多大？我们封装了关键字Create Account来创建用户，现在看样子只需要把关键字的实现部分中调用Create命令的地方改成Register就可以了，但这样一来的又一个问题就在于我们的测试的关键字和被测的功能也出现了不一致的叫法，这会使人困惑。或许以后我们每次验收测试跑完之后，都需要向这些测试报告的经理或者市场人

员解释这些关键词。

为了保持术语一致性，最好的做法是修改我们的测试用例。上面的测试用例中，至少有八个用到Create Account关键字的地方可能都要改成Register。还有两个关键字也都用到了Create Account。而现实会更残酷，可能有成千上万的测试步骤都调用了那个关键字。由此，我们得出结论：重复绝对会增加维护成本。

重复往往预示了潜藏在测试中的某一重要概念。当这样的重复不是发生在个别测试步骤而是一系列的步骤的时候，情况更是这样。

思考一下表3中的测试脚本，看看前面两行究竟说明什么。没错，它们核实创建用户命令是否拒绝1234!@$^这一密码。那再来看看第9~10两行。这两行来证实创建用户命令接受!C2456这一密码，再进行一次抽象概括，我们惊奇地发现，原来这两行测试的本质便是接受（Accept）和拒绝（Reject）。但遗憾的是在表3的测试中，这一本质却被埋没了。接下来我们利用两个新的关键字来使概念明朗化，如表4所示：

表4 用两个新的关键字来使概念明朗化

Listing 4: Keywords for accepting and rejecting passwords

```
** Keywords **
Accepts Password ${valid_password}
    Create Account arbitraryUserName ${valid_password}
    Status Should Be Account Created

Rejects Password ${invalid_password}
    Create Account arbitraryUserName ${invalid_password}
    Status Should Be Invalid Password
```

这两个新的关键字不仅将重写我们的测试脚本，同时也给接受密码和拒绝密码下了定义：接受密码即调用被测系统的Create命令，系统报告用户创建成功；拒绝密码即调用Create命令，系统报告密码无效。

如此一来再次经过修改的测试脚本如表5所示，减少了重复并且更能体现被测功能的职责，即密码的接受或拒绝。

让我们捋一捋刚才的思路，总结一下该部分。首先我们经过分析测试代码的重复部分，发现被测功能的两个最基本概念——接受正确的密码和拒绝错误的密码。通过定义两个关键字，我们抽象并命名了这两个基本概念。最后我们在测试用例中使用新的关键字，从而提升了测试的可读性以及可维护性。

表5 减少了重复的重写测试

Listing 5: Test rewritten to reduce duplication

```
** Test Cases **
The create command validates passwords
    Rejects Password 1234!@$^
    Rejects Password abcd!@$^
    Rejects Password abcd1234
    Rejects Password !2c45
    Accepts Password !2c456
    Accepts Password !2c4567890123456
    Rejects Password !2c45678901234567
```

给本质一个有意义的名字

经过一番“折腾”，现在表5给出测试已经能够比较清晰地表达测试的基本概念。而与此同时，最后一点不够清晰的地方已经开始明朗起来。看看测试中的几个密码，我们不能马上弄清到底给出的密码无效在什么地方？那正确的密码是符合了什么样的规则吗？或许花些时间你就可以找到问题的答案。这里涉及一个重点：任何花在思考测试的意义即其本质的时间都算作维护成本 。这个成本看起来微不足道，但是如果系统需求更改导致大面积关联的测试用例都要修改，这时累加起来的成本是巨大的。我的一些客户已经发现这个问题的严重性，道理很明白，千里之堤，溃于蚁穴。

在上述的测试例子中，我所选取的每一个密码 都有特定的目的，也就说每一密码的本质都是和被测系统功能的某一个需求有关。比如1234!@$^这个密码，它不含字母， 因而这个密码的本质就可以这么描述：一个不包含字母的密码。

我习惯给每一个本质赋予具有意义的名称，在测试代码中给本质命名的是变量。有时候我也创建变量，给它命名一个富有表现力的名字，再给它规定一种能体现其名的价值。如下，我定义了一个变量用来储存没有字母的密码。

然后在测试脚本中使用变量，节省空间，这里略去了其它变量的定义过程，如表6所示，每一个密码都以变量的形式表达其代表的本质意义。至此，离我们开始设定的目标已经很近了，但依然有可以再优化的地方，我将进一步把原来一个测试按照密码不同属性拆分成多个测试用例，如表7。

现在，只需看一眼便可知每一个测试用例

表6 在测试脚本中使用变量，节省空间

Listing 6: Test rewritten to name significant values

```
** Test Cases **
The create command validates passwords
    Rejects Password ${aPasswordWithNoLetters}
    Rejects Password ${aPasswordWithNoDigits}
    Rejects Password ${aPasswordWithNoPunctuation}
    Rejects Password ${aTooShortPassword}
    Accepts Password ${aMinimumLengthPassword}
    Accepts Password ${aMaximumLengthPassword}
    Rejects Password ${aTooLongPassword}
```

表7 把原来一个测试按照密码不同属性拆分成多个测试用例

Listing 7: Test rewritten to name significant values

```
** Test Cases **
Rejects passwords that omit required character types
    Rejects Password    ${aPasswordWithNoLetters}
    Rejects Password    ${aPasswordWithNoDigits}
    Rejects Password    ${aPasswordWithNoPunctuation}

Rejects passwords with bad lengths
    Rejects Password    ${aTooShortPassword}
    Rejects Password    ${aTooLongPassword}

Accepts minimum and maximum length passwords
    Accepts Password    ${aMinimumLengthPassword}
    Accepts Password    ${aMaximumLengthPassword}
```

或者每一步的意义何在。这里，重要的需求概念被清晰精炼地表述了出来。

现在假定新需求改变了密码极限长度，由于每一个需求和被测功能都由测试用例清晰地显示出来，我能够很快定位哪一个测试用例需要修改。并且由于每一个测试数据也即密码都以有意义的变量的形式储存，我们能够很快的找到需要修改的变量进行重新赋值。先前对测试代码所做的重构带来的好处显而易见，这里再一次强调测试即开发的主旨。

让测试经得起测试：应对系统主要实现架构的改变

在前面部分我们努力让测试能够更灵活地自适应需求变更，可如果是系统的主要实现架构发生了变化呢？测试会受到什么影响？为了找出答案，我们通过改变一点实现的细节——通过Web页面创建用户的方式取代之前的命令行调用Create命令的方式。现在创建用户只需要打开创建用户的Web页面，输入用户名和密码，然后点击创建用户按钮，由页面打印创建结果。严峻的问题来了，我们的测试该如何修改？

还记得我们之前封装不必要细节并提炼了两个关键字Create Account和Status Should Be。这两个关键字封装的细节就是如何调用命令行执行Create命令以及获得结果报告。很明显，我们肯定要重写这两个地方，因为现在需要通过Web页面操作才能实现。

表8 修改后的关键字实现

Listing 8: Rewriting keywords to invoke the new web app

```
Create Account ${username} ${password}
    Go To    http://localhost:4567/create
    Input Text    username    ${username}
    Input Text    password    ${password}
    Submit Form

Status Should Be ${required_status}
    ${status}=          Get Text    status
    Should Be Equal     ${required_status}  ${status}
```

表8是修改后的关键字实现。我们修改了与系统交互的实现，即通过使用开源Web测试工具Selenium进行页面访问和操作，那么现在的问题是对于那几个具体的自动化测试用例，我们还需要修改什么？答案是什么都不需要，此次修改工作已完毕。通过改变几行代码，使我们的自动化测试轻松运行在变化了的实现架构的系统上，这往往就是成功和失败的自动化测试之间的区别。

与此同时，回到现实世界

在真实的测试中，你可能要做更多的工作以应对系统实现架构的变化，你可能不止需要修改两个关键字。但只要你创建了级别较低的关键字，将其他代码和与系统交互的细节分开， 那么你所需要做的就只是修改这些关键字而该测试用例照常运行不需要改动【译者注：如果你看到Martin Flower的《重构》一书，就应该明白这样一条重构原则，保持接口不变，改变底层实现】。

真实的项目里很多实现架构上的变动将会对测试开发工具提出更严酷更颠覆的问题。 但即使是最坏情况，你依然可以使用先进的开源测试工具，用以解决很多重复的问题，帮助你撰写能够清晰表达测试本质的用例和脚本。

再次强调，一定要记住其中的本质内容：只有消灭重复的无关紧要细节，让测试清晰表达被测系统功能职责，才能在发生系统需求和实现变更的情况下轻松应对以便降低自动化测试维护的成本。这也正是我们衡量自动化测试开发成功的标志。

责任编辑：高松（gaosong@csdn.net）



SDK安装程序卡壳之谜

兼谈函数的异常出口（上）

2005年，我做了一场题为《C++异常处理得与失》的演讲，引起了激烈的争论，形成了“赞成抛异常”和“反对抛异常”两个派别，争论持续到演讲结束。我当时的观点是倾向哪一方呢？看下这篇文章吧，看后自然晓。

主持人：张银奎

《软件调试》一书作者，从事软件开发和研究10余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。翻译（合译）作品包括《数据挖掘原理》、《机器学习》、《人工智能：复杂问题求解的结构和策略》、《观止——微软创建NT和未来的夺命狂奔》等。

安装SDK

代码是软件的根本，写代码就像写文章一样，是值得修炼一生的技艺，提高对代码的感知力和控制力是无止境的。因此，我还经常写代码，机器上常备着几种开发环境，比如开发Windows驱动程序的DDK/WDK，开发Windows应用程序的Visual Studio 2010和Platform SDK，开发Linux驱动和应用程序的Ubuntu虚拟机，开发Java程序的Eclipse，还有钟爱多年的Visual C++ 6.0（VC6）。安装这些工具一般不需要费什么事，借助今天高速便捷的网络环境，通常把安装程序启动起来，然后起身出去倒杯茶或者散散步，回来之后便安装好了，比起当年学编程时使用8张软盘安装Borland C++ 3.1真是今非昔比。但也有不顺利的时候，尤其是最近几年开发工具也普遍患上了肥胖症后，开发工具的安装程序也变得复杂起来，有时也会卡壳。今天要给大家讲的就是本人亲历的Windows 7 SDK安装程序陷入僵局的故事。

```
0:009> ~*e ? @@tid; !gle
Evaluate expression: 2132 = 00000000`00000854
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0xc0000034 - Object Name not found.
Evaluate expression: 6080 = 00000000`000017c0
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0 - STATUS_WAIT_0
Evaluate expression: 9824 = 00000000`00002660
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0 - STATUS_WAIT_0
Evaluate expression: 8604 = 00000000`0000219c
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0 - STATUS_WAIT_0
Evaluate expression: 8448 = 00000000`00002100
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0 - STATUS_WAIT_0
Evaluate expression: 2460 = 00000000`0000099c
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0 - STATUS_WAIT_0
Evaluate expression: 5860 = 00000000`00001664
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0xc000008a - Indicates the specified resource type cannot be found in the image file.
Evaluate expression: 5272 = 00000000`00001498
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0xc0000034 - Object Name not found.
Evaluate expression: 4620 = 00000000`0000120c
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0xc000007c - An attempt was made to reference a token that doesn't exist.  This is ty
Evaluate expression: 5336 = 00000000`000014d8
LastErrorValue: (Win32) 0x3 (3) - The system cannot find the path specified.
LastStatusValue: (NTSTATUS) 0xc000003a - {Path Not Found}  The path %hs does not exist.
Evaluate expression: 9224 = 00000000`00002408
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0 - STATUS_WAIT_0
Evaluate expression: 4484 = 00000000`00001184
LastErrorValue: (Win32) 0 (0) - The operation completed successfully.
LastStatusValue: (NTSTATUS) 0 - STATUS_WAIT_0
```

图1 观察线程的Last Error

记得当时是从一个网络共享目录启动安装程序，也就是执行名为Setup.exe的可执行程序。在确认程序显示出了安装界面并开始忙碌后，我便按老规矩站起身出去倒茶。倒茶回来后，看到的是从显示信息看是在下载文件，喝茶等会儿吧。但过了几分钟后，仍然没有什么进展，进度条丝毫未动。根据多年的经验，一定是哪里出什么问题 。因为SDK安装程序算不上是很有影响力的软件，没有太大的吸引力让我动调试器，所以当第一次遇到这个现象时，我选择的是IT同行惯用的手法——重试一次（Try Again），也就是Ctrl + Shift + Esc调出任务管理器，将停滞的安装程序进程杀掉，然后再重新运行。这次没有去倒茶，因为有会议要开。等到一个小时的会议开好了后，回来一看，还是停在没有进展的状态。这样连续出错三次后，我只好上调试器了。

上调试器

唤出WinDBG，附加到名为SDKSetup的进程。这里有必要说明一下，前面曾提到，开始安装时执行的是Setup.exe，这里为什么要附加到SDKSetup进程呢？原因是前者只是一个很轻量级的前导程序，并不负责具体的安装任务，它的主要职责就是以管理员身份启动后者。另外要指出的是，当时安装的是64位版本的SDK，SDKSetup进程是64位进程，因此使用的WinDBG也是64位的。



图2 可疑线程的栈回溯

接下来的问题是如何找突破口。经常看本专栏的朋友一定已经有了些思路。但这个问题不同于一般的GUI挂死（窗口失去响应），如果观察0号线程，应该看不出什么反常。长话短说，对于这个问题，观察每个线程的Last Error很有效，也就是执行如下命令：

```
~*e ? @$tid;!gle
```

也就是显示进程中所有线程的线程ID和Last Error（GetLastError() API所获取的值），执行结果如图1所示。

在图中，每个线程的信息占3行。以0号线程为例，它的线程ID是0x854，LastErrorValue为0，表示成功结果，LastStatusValue为0xc0000034，代表了错误情况。这两个值都保存在线程的TEB结构中，Windows API的内部代码会维护这两个变量的值。

```
0:000> dt _TEB -y Last
ntdll!_TEB
+0x068 LastErrorValue : Uint4B
+0x1250 LastStatusValue : Uint4B
```

比如下面的栈回溯反映的就是打开注册表键（RegOpenKeyExW）时失败而更新LastStatusValue的值。

```
00 ntdll!RtlNtStatusToDosError+0x1c
01 kernel32!LocalBaseRegOpenKey+0x230
02 kernel32!RegOpenKeyExInternalW+0x123
03 kernel32!RegOpenKeyExW+0x1d
```

为什么LastErrorValue代表的是成功而LastStatusValue代表的是失败呢？原因是有些API只设置LastErrorValue，不设置LastStatusValue。目前的情况应该是LastErrorValue被后来调用的API恢复成0了。这意味着，0号线程曾经遇到过调用API失败的情况，LastStatusValue被设置成了失败值，但最近调用的API是成功的。LastErrorValue被恢复成0，但这个API没有恢复LastStatusValue。因此，我们暂时放过0号线程，继续寻找最近一次调用API失败的线程，也就是LastErrorValue值不为0的线程。按这个标准来筛选，图2中的12个线程中，只有14d8号线程符合标准。

它的LastErrorValue为3，意思是系统找不到指定的路径（The system cannot find the path specified.），它的LastStatusValue为0xc000003a，意思也是路径不存在（{Path Not Found} The path %hs does not exist.）。两个“证据”相互印证，增加了可信度。我们是否抓到了“狐狸尾巴”呢？

```
Call Site
SDKSetup.DownloadUtil.Dispose(Boolean)
SDKSetup.DownloadThread.DownloadThreadProc()
System.IO.__Error.WinIOError(Int32, System.String)
System.IO.FileInfo.get_Length()
SDKSetup.DownloadUtil.DownloadFileAsyncUsingAsyncFileIO(System.Strin
SDKSetup.DownloadUtil.DownloadFileAsync(System.Uri, System.Uri)
SDKSetup.DownloadUtil.DownloadFile(System.Uri, System.Uri)
SDKSetup.DownloadThread.DownloadThreadProc()
System.Threading.ExecutionContext.Run(System.Threading.ExecutionCont
System.Threading.ThreadHelper.ThreadStart()
```

图3 SOS显示的托管栈回溯

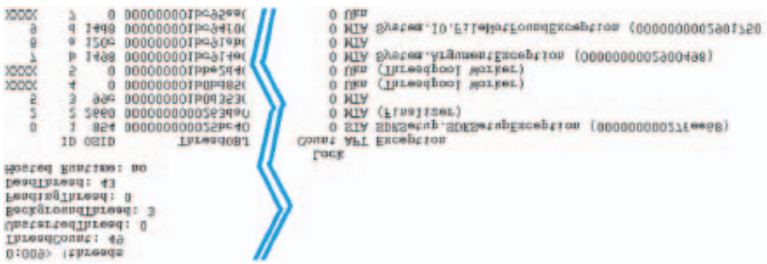


图4 使用SOS观察线程信息

细看14d8号线程.

14d8是操作系统分配的线程ID，在WinDBG中切换线程通常使用的是WinDBG分配的线程序号，如果要使用线程ID，那么可以使用下面这样的命令：

```
~~[14d8]s
```

如果觉得这个命令的语法啰嗦，那么也可以从上至下数一下，可以知道14d8是9号线程，执行~9s来切换而达到同样的效果。

如何快速了解这个线程呢？看它的“履历”，也就是执行k 100命令看它的执行经过，如图3所示。首先看图2中顶部的5个栈帧，据此可以推测这个线程目前在调用Sleep函数“打瞌睡”。对于分析这样的卡壳问题，看到有人在调用Sleep函数，可以说是一个“好”消息。

接下来的两个栈帧无法解析到任何模块，

结合栈中多次出现的.NET运行时（mscorwks）模块，可以知道它们对应的是.NET及时编译（JIT）所产生的代码。这充分说明，这个SDK安装程序使用了.NET编程技术。再往下看，可以看到抛出异常（JIT_Throw）、分发异常和处理异常的一些函数，有了这样一个初步印象后，姑且放下从本地代码角度的分析。因为这是个托管（.NET）线程，因此加载我们曾经介绍过的SOS扩展来帮忙。

细看现场

执行.loadby sos mscorwks命令，从mscorwks模块所在的位置加载SOS扩展，然后执行!clrstack观察托管方法的调用经过，如图4所示（去掉了Child-SP 和RetAddr两列）。

细看图3，多么完美的栈回溯啊！非常清楚地显示了这个线程的执行经过。由下至上，最下面两个栈帧是线程的启动函数。接下来是DownloadThread类的DownloadThreadProc方法，显然是这个线程类的线程入口，从类名和方法名都可以看出这个类的用途是下载文件。而后是DownloadUtil对象的DownloadFile方法，目的是下载文件，这个方法又调用同一个类中的DownloadFileAsync方法，目的是以异步方式下载文件，而后又调用的是DownloadFileAsyncUsingAsyncFileIO，意思是使用异步文件IO的方式来下载文件。

下载文件操作被这样层层“外包”了几次后，DownloadFileAsyncUsingAsyncFileIO终于要做点实事了，也就是调用IO函数取文件长度，System.IO.FileInfo.get_Length。接下来明显是报告错误__Error.WinIOError，事实上，也正是这个函数内抛出了一枚炸弹，也就是throw了一个System.IO.FileNotFoundException异常。使用SOS的!threads命令可以看到这个异常的对象地址（9号线程），如图4所示（局部）。

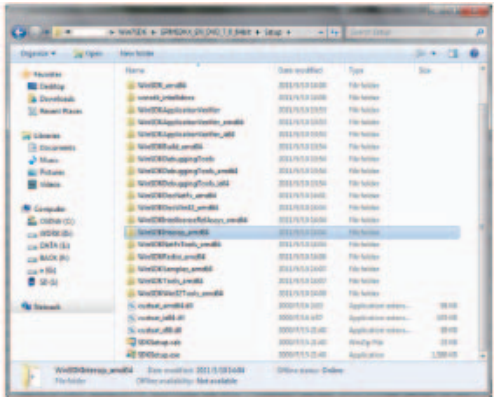


图5 查看安装文件

有了异常对象地址后，可以使用!do（dumpobject）命令来显示此异常对象的详情：

```
0:009> !do 0000000002901750
Name: System.IO.FileNotFoundException
MethodTable: 000007fef88b4c78
EEClass: 000007fef84a7860
Size: 160(0xa0) bytes
Fields:
…
000… System.String   0 instance
0000000002901570 _message
       …
```

再使用!do命令显示_message字段的值，就可以看到这个异常的文字描述了：

```
0:009> !do 0000000002901570
Name: System.String
MethodTable: 000007fef8897a80
EEClass: 000007fef849e530
Size: 478(0x1de) bytes (C:\Windows\
assembly\GAC_64\...\mscorlib.dll)
String: Could not find file
'\\filesrv722\DevTools\Win7SDK\...
_7_0_64bit\Setup\WinSDKInterop_ia64\cab1.
cab'.
```

哦，原来是找不到这样一个文件。这与前面的Last Error信息很是一致。是不是这个文件真的不存在呢？手工浏览安装文件所在的目录（图5），真的没有这个目录！我的机器是x64平台，因此选用的是x64版本的SDK，但从上面的文件路径来看，寻找的子目录以ia64结束，代表的是安腾64（Itanium），岂不谬哉？

讲到这里我们知道触发问题的直接原因是下载文件的线程在试图下载一个不存在的文件时触发了异常。但故事还没有结束，找不到一个文件也不该就卡住不动啊？最好给个错误信息啊，不然崩溃了产生转储也比卡住不动强啊。从这个角度来看，程序的代码逻辑可能也存在不足之处。从图2看，有人抛出了异常，但程序并没有崩溃。从本文开头描述的症状看有点像是死循环了，结合图3所示的栈回溯，可能是在那里反复调用Sleep。如果真的是这样，那么CPU又是如何陷入到这个死循环中的呢？因为篇幅关系，我们下一期继续分析。

责任编辑：高松（gaosong@csdn.net）

本期问题：

上期答案是：上一期的问题是Android的DDMS（Dalvik Debug Monitor Service）工具还有哪些辅助调试或调优的功能。答案是还有接收和显示日志信息的LogCat功能，辅助调试的触发GC回收功能和屏幕截图功能（Screen Capture），以及供调优使用的HPROF功能和Method Profiling功能。HPROF是Java中的一种调优技术，常用于收集堆和CPU的使用信息。Method Profiling功能可以追踪每个Java方法的调用次数和执行时间。

本期问题是：你知道GetLastError() API是如何访问到TEB结构中的LastErrorValue字段的吗？

编者说明：

- 投稿信箱：contest@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
 1）姓名
 2）工作单位或学校
 3）电话联系方式
 4）邮寄地址
 5）E-mail地址
- 解答提交时间，最好早于当月15日。

Y X-RIME：基于Hadoop的开源大规模社交网络分析工具

文 / 陈冠诚，史巨伟，杨博，杨寅

X-RIME是一个基于Hadoop的开源社会网络分析工具。依赖于Hadoop提供的大规模数据并行处理能力，实现了对十几种网络分析算法的并行化，提供了一整套用于对大规模社会网络进行分析处理的解决方案。本文对X-RIME进行了介绍。

随着互联网的快速发展，涌现出了一大批以Facebook、Twitter、人人和微博等为代表的新型社交网站。这些网站用户数量的迅速增长使得海量的用户数据不断被产生出来，而如何有效地对这些海量的用户数据进行社交网络分析（Social Network Analysis）正成为一个越来越热门的问题。本文向大家介绍由IBM中国研究院和北京邮电大学合作开发的X-RIME开源库（http://xrime.sourceforge.net/），一个基于Hadoop的开源社交网络分析工具。

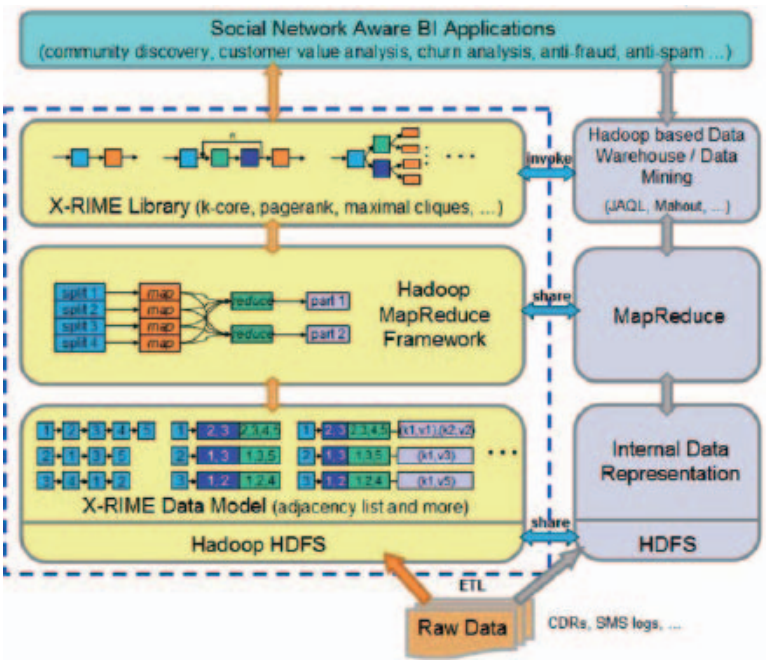


图1 X-RIME整体架构

其实早在20世纪90年代初就已经有许多企业和研究机构对社交网络进行过相关研究。然而随着互联网用户的急速的增长，今日的社交网站所需处理的数据已经不是传统的解决方案所能够应对的了。例如，传统的社会网络分析算法和工具往往都是单机形式的，在面对大规模数据集时往往会出现存储和处理能力不足等方面问题，再加上原始输入数据和社会网络的内部表示大都属于无结构或者半结构化数据，传统关系数据库并不擅长处理此类数据，使得利用传统的社会网络分析算法和工具对大规模数据集进行处理变得更加困难。另一方面，随着Hadoop的日益流行，许多中小互联网企业可以通过搭建Hadoop集群来方便地进行大规模数据处理。然而，Hadoop并不直接提供社交网络分析的算法库，因此实施海量社交网络分析仍存在较高门槛。基于这些需求，我们设计并实现了X-RIME。

X-RIME是基于Hadoop的开源社会网络分析工具。依赖Hadoop提供的大规模数据并行处理能力，X-RIME实现了对十几中网络分析算法的并行化，提供了整套用于对大规模社会网络分析处理的解决方案。通过使用X-RIME，用户可以方便快捷地对海量社会网络数据进行分析，从这些海量社会网络数据中获取更深层次的有用信息，从而进一步挖掘商业价值，支持商业决策以及发现新的业务增长点。

X-RIME架构介绍

图1描述了X-RIME的整体架构，它主要由四层组成：HDFS、X-RIME数据模型、X-RIME算法库以及基于社交网络分析的商业智能分析应用。X-RIME算法库是X-RIME的核心组成部分，它基于Map/Reduce实现了十余种分布式社交网络处理算法。

X-RIME最底层采用了HDFS来存储海量数据。像很多其他基于Hadoop的数据分析解决方案一样，X-RIME也采用了HDFS来构建底层的海量数据存储设施。整个X-RIME算法库的所有输入文件、中间结果和最终结果都会存储在HDFS上。

处于倒数第二层的X-RIME数据模型层实现了社交网络数据的“数据结构”。我们知道，社交网络的基础模型是图论中的图模型。在这个模型中，社会网络的个体被视为图中的节点，个体之间的关联被视为图中的边。X-RIME数据模型层包括了近20种数据结构，主要包括基于Hadoop的对社会网络中的点、边等抽象概念的具体数据结构表示。在后面一节我们会详细介绍该数据模型的设计原则。

在X-RIME数据模型层之上的是X-RIME核心算法库（它运行在Hadoop的MapReduce框架之上）。在算法库中，我们通过map()/reduce()函数对的形式实现了十余种常见的社交网络分析算法。这些算法通过将多个Hadoop Job按算法工作流程组合在一起来共同完成相应的任务。这些算法都被相同的接口封装起来，这些接口一般包括四种参数：第一，输入文件在HDFS中的路径，它保存了与X-RIME数据模型相兼容的输入文件；第二输出文件在HDFS中的路径，它用以保存最终的分析结果；第三，Map/Reduce的相关参数，例如Mapper数或者Reducer数等；第四，社交网络分析算法相关参数，例如迭代次数等。

图1中最顶层是基于社交网络分析的商业智能分析应用。它通过调用X-RIME核心算法库来实现对社交网络的数据分析。如果需要的话，用户还能将它与已有的数据仓库解决方案集成（例如JAQL、Mahout等），从而提供一个更加完整、高效的综合商业智能分析解决方案。

X-RIME 数据模型的设计原则

X-RIME的设计目标是用来专门做大规模数据集社会网络分析的工具，因此我们对X-RIME数据模型进行设计时必须考虑以下两点原则：X-RIME需要处理大规模数据集；X-RIME分析的对象是社会网络。X-RIME处理大规模数据集的能力主要依赖于Hadoop的大规模并行处理能力，因此只要X-RIME中所有的数据结构都是基于Hadoop的海量数据集接口即可。这里我们重点分析X-RIME分析的对象即社会网络的特点。之前的分析中已经提到社会网络的基础模型是图论中的图模型，在这个模型里，社会网络中的个体被视为图里的结点v，结点的集合为V；个体之间的关联被视为图里面的边e，边的集合是E={e (u, v) | u ∈ V, v ∈ V}，因此整个模型就可以看作是G=(V, E)。基于此我们对X-RIME的数据模型做了如下考量。

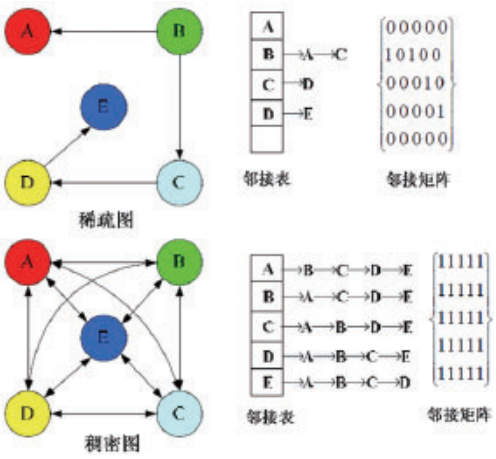


图2 稀疏图和稠密图的邻接表与邻接矩阵形式

采用邻接矩阵还是邻接表

如图2所示，要表示一个图G = (V, E)，有两种标准的方法，即邻接矩阵和邻接表。一般认为当|E|远小于|V|2的图属于稀疏图，反之则认为是稠密图。使用邻接矩阵表示法的优点在于可以很快判断两个给定结点是否存在连接边，缺点在于当要表示的图是稠密图的时候有大量的空间会被浪费。邻接表表示方式的优

点在于节省空间，缺点在于判断两个给定结点是否存在连接表需要遍历其中某个结点的邻接表，效率较低。基于以下两点考虑，我们采用了邻接表的方式表示X-RIME中的图结构。

- 社交网络一般属于稀疏图结构，因此使用邻接表表示可以节省大量空间，提高空间利用率。
- X-RIME中大部分算法不需要快速判断两个给定结点是否存在连接边。

边的表现形式

在邻接表中，结点之间的关系需要使用边来承载，边的形式可以有多种，如有向边、无向边、自环边（自己指向自己）等。

考虑到在社会网络中，上述几种边都有可能存在，在不同的应用场景中有不同需求，因此我们需要有灵活的数据结构来支持上述各种不同形式的边。

此外还有一种情况需要考虑，当有向边用{from, to}来表示时，传统的邻接表表示法只是将这条边信息记录在from端，但在社会网络分析中，我们可能存在某种场景需要同时将这条边信息记录在to端，X-RIME的设计中考虑了这种应用场景。

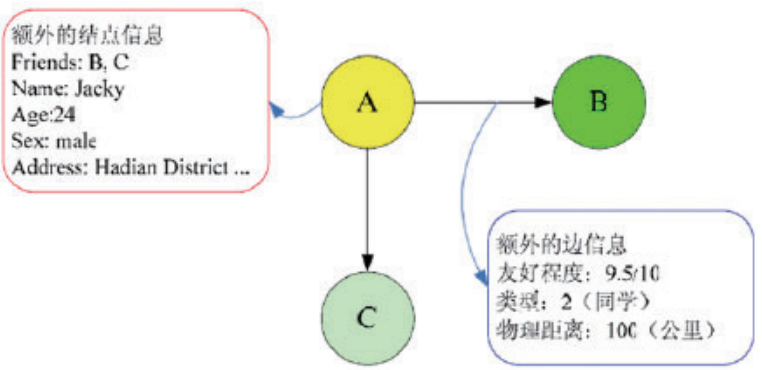


图3　社会网络中结点和边需要存储额外信息

额外的承载信息

X-RIME需要处理的社会网络图与传统的简单图不一样，它是个体以及个体之间复杂关系的一种抽象。

如图3所示，在社会网络中，结点自身往往需要存储一些额外的信息，例如当图中的结点表示人时，可能需要额外记录这个人的性别、年龄、家庭地址等信息；结点之间的关系（边）往往也需要存储一些额外的信息，例如当图中的边表示两个人是好朋友时，可能需要额外记录这条边的强度（好友关系的强烈程度）、边的类型（关系类型，如家人、朋友、同学等）、好友间的物理距离等。基于上述考虑，X-RIME的设计中必须考虑为结点和边提供额外的信息存储功能。

比较器

在社会网络中，个体和边需要进行某种程度的对比。例如在好友关系网中，人们可能希望比较得出哪些人是自己最好的朋友，人们同样可能希望比较得出自己在好友心目中的重要程度等。映射到X-RIME中，大量的运算的确需要对结点以及边进行比较。这种比较可以是简单的数值比较（例如边的权值比较）也可以是复杂的逻辑比较（例如综合边的关系类型、边的强度、结点之间的物理距离等进行比较）。X-RIME的设计中必须考虑数据类型之间的比较，需要设计各种比较器。

效率问题

X-RIME需要处理的是大规模海量数据，如果我们对输入数据的读写处理只是简单地根据原始的文本文件格式进行读写，势必影响效率，因为这样多了一个中间转换过程，需要读入内存再根据特定的数据结构格式进行转换。Hadoop提供的序列化I/O接口为我们提供了一个有效的方法来提高读写效率。在读取输入数据之前，我们需要预先对原始文本进行转换，通过Hadoop序列化I/O接口的序列化功能将其转换成二进制镜像文件形式，这样每次X-RIME读取被序列化产生的二进制文件时可以直接通过Hadoop序列化I/O接口的反序列化功能将镜像文件装载到内存里，输出时直接通过Hadoop I/O的序列化功能进行输出，效率大大提高。两种读写方式如图4 所示。

X-RIME使用介绍

使用X-RIME大致可以分为四步。

第一步：获取原始数据，例如使用爬虫获

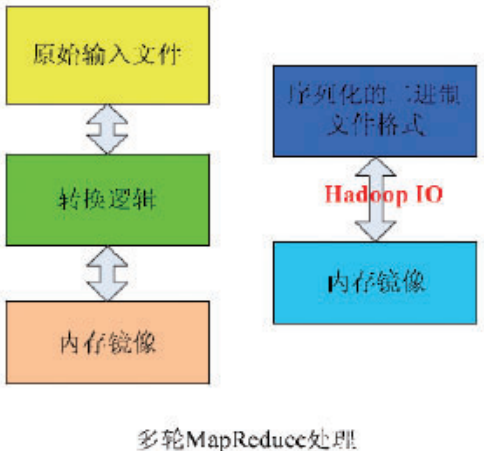


图4 两种输入输出方式
（左：较为低效的传统方式，右：高效的序列化方式）

取原始网站数据。

第二步：对数据进行预处理以转化成X-RIME数据模型所支持的格式。这个步骤与用户提供的具体数据格式相关，因而通常由X-RIME用户自己实现。

第三步：调用X-RIME算法库对这些数据进行社交网络分析。

第四步：对X-RIME的输出结果进行整合，生成易于理解的文档。

下面我们来介绍下使用X-RIME对某BBS中一个分论坛进行弱连通分支（Weakly Connected Components，后面简称WCC）算法分析的结果。在BBS中，每一个帖子的发起者A是一个节点，而如果另一个用户B回复了这个帖子，我们说这两个用户间形成了一个关系，即B指向了A。

图5中的蓝红紫三条线分别代表该BBS中MilitaryView版、Circuit版和Career_POST版的WCC分布情况。从图中我们可以看到，

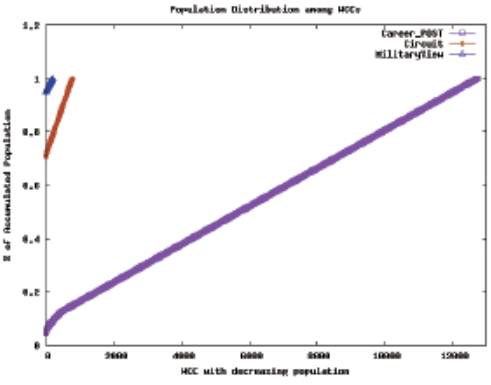


图5 弱连通分布

MilitaryView版和Circuit版中大部分用户的WCC值都很高。这说明这两个版块中的大部分用户彼此都直接或者间接的联系在一起。相反的，Career_POST版中大部分的用户彼此间的联系都非常松散。其实这个结果非常易于理解，因为MilitaryView和Circuit版是专门的版块，在这个版块的用户大都是基于相同的兴趣而产生的发帖、回帖行为，因此彼此间的互动更频繁、联系更紧密；相对的，Career_POST版主要被用于发布和浏览招聘信息，因此用户的回帖行为不多，用户间的关联性不强。

总结

X-RIME作为基于Hadoop的开源工具，为大家提供了一种方便快捷地进行大规模社交网络分析的新选择。如果你对X-RIME有什么新的需求或者建议，欢迎你直接与我们联系。

参考文献

[1] X-RIME Homepage: http://xrime.sourceforge.net/

[2] Wei Xue, JuWei Shi, Bo Yang. X-RIME: Cloud-Based Large Scale Social Network Analysis. Proceedings of 2010 IEEE International Conference on Services Computing.

[3] Kai Shuang, Yin Yang, Bin Cai, Zhe Xiang. X-RIME: HADOOP-BASED LARGE-SCALE SOCIAL NETWORK ANALYSIS. Proceedings of IC-BNMT2010.

[4] 杨寅.大规模社会网络分析数据模型的设计与实现. 中国科技论文在线.

作者简介

陈冠诚：IBM中国研究院研究员；史巨伟：IBM中国研究院研究员；杨博：IBM中国研究院资深研究员；杨寅：就职于人民搜索网络股份公司。

新书上架

本月新书



互联网创业启示录

作者：Bob Walsh

译者：曹铁鸥

本书涉及网站创业的现状和机遇、创业公司价值、平台选择、工具和群组、资金的筹集、管理和盈利、社会化媒体环境、行动执行管理、开发人员方法论和工作效率、创始人角色等，内容既有纲领和指导性理论，又有具体操作方法。



Oracle SQL高级编程

作者：Morton,K.

译者：朱浩波

Karen Morton及团队在此书中提供了专业的方案：先掌握语言特性，再学习Oracle为提升语言效率而加入的支持特性，进而将两者综合考虑并在工作中加以应用。



精通Android 3

作者：Satya Komatineni Sayed Hashimi Dave MacLean

译者：杨越

作者在上一版的基础上对本书做了全面改进，增加了Android内部构件的知识，介绍线程、进程、长期运行的服务、广播接收程序和闹钟管理器。还介绍了更多UI控件，并用150多页的篇幅专门介绍了Android 3.0版，包括碎片、碎片对话框、ActionBar和拖放。



实战Windows Azure：微软云计算平台技术详解

作者：徐子岩

本书主要包括Windows Azure平台的基础知识、使用方法、功能特点以及如何基于Windows Azure平台设计高可靠、可扩展的应用程序，涵盖Windows Azure平台所有内容。可从中学到Windows Azure平台都提供了哪些云计算的功能、应用程序如何使用这些功能，从而构建一个高可用、可伸缩、高性能的应用程序。

推荐图书



RESTful Web Services Cookbook中文版（REST最佳实践手册）

作者：Subbu Allamaraju

译者：丁雪丰 常可

出版社：电子工业出版社

Web取得了巨大的成功，其技术层面的原因究竟是什么？

经过深入细致地分析之后，人们将Web取得成功在技术层面的原因总结为一种崭新的分布式应用架构风格——REST（表述性状态转移）。简单来说，Web的架构风格就是REST，所有希望在Web生态系统中繁荣发展的技术。

HTTP1.1规范正是基于REST架构风格的指导原理来设计的，其目标是实现符合REST风格的分布式应用架构。HTTP1.1规范在1999年6月正式发布。次年，HTTP1.1规范的主要设计者Roy Fielding在他的博士论文*Architectural Styles and the Design Of Network—basedS of tware Architectures*中，系统地阐述了REST这种架构风格的推导过程和详细理念。

2005年，Web 2.0崛起。

这场运动受到了Ruby on Rails的大力推动。Rails的创造者DHH曾经介绍了在Rails1.2版中对于REST开发的支持，并且将支持REST开发作为Rails未来发展的优先目标。DHH在Web开发者社区中强大的营销能力，大力推动了REST开发的普及。尽管如此，Rails所支持的REST开发却将对资源的操作局限于CRUD（创建、读取、修改、删除）的语义（即：将对资源的CRUD操作映射到GET、POST、PUT、DELETE四个HTTP方法），这其实是收窄了REST的适用范围。而且在Rails对于REST开发的支持中，并没有直接支持超文本驱动的理念，而这是REST架构风格最核心的理念，也是REST风格的架构达到松耦合目标的根本原因。可以这样说，Rails对于REST开发的支持方式，给Web开发者社区带来了很大的误导。其他编程语言的Web开发框架（例如Java语言的Struts、Spring MVC等）对于REST开发的支持，大多模仿了Rails的方式，这导致了一线的Web开发者误以为：REST开发就是通过GET、POST、PUT、DELETE四个HTTP方法对资源执行CRUD操作。但随着Web开发者社区逐步深入实践REST开发，对REST的认识也逐渐深入，并将REST API必须是超文本驱动的当作共识。

与此同时，REST与其他来自Web应用的技术（HTTP、HTML、CSS、JavaScript、AJAX、Atom）一起，逐渐渗透进了企业应用的领域。一些企业应用（特别是SOA）的架构师们发现，REST架构风格对于实现SOA的目标具有非常大的价值。IBM、Oracle（包括BEA+Sun）、Microsoft等大公司的企业应用产品中有很多都已经提供了REST风格的API。与此同时，大量的开源项目和云计算服务都提供了REST API接口。

作为O'Reilly出版的Cookbooks，本书风格言简意赅。作者不曾设想手把手地教会读者如何实践，而是把目标读者设定为系统架构师，他们应该已经知道正在做什么，但需要了解使用RESTful时自己的目标是什么。书中给出的方法指出了如何实现读者的目标，因而更像是最佳实践。

对比其他论述REST的书籍，本书极力强调使用一致的XML格式，这也是阅读此书获得的最重要的启发之一。只使用PUT和DELETE，对于想真正理解representations是远远不够的，还必须设计有意义的XML，沿着这些思路，作者关于Atom协议的讨论特别引人入胜。



黑客：计算机革命的英雄
作者：Steven Levy
译者：赵俐　刁海鹏　田俊静
出版社：机械工业出版社

Steven Levy这部经典的25周年版从20世纪50年代早期跨越到80年代后期，追述了计算机革命中初期黑客的丰功伟绩。Hackers都是最聪明和最富有个性的精英，他们勇于承担风险，勇于挑战规则，并把世界推向了一个全新的发展方向。本书更新了一些著名黑客的最新资料，包括比尔·盖茨、马克·扎克伯格、理查德·斯托曼和史蒂夫·沃兹尼亚克，并讲述了从早期计算机研究实验室到最初的家用计算机期间发生的的故事。

在Levy的笔下，他们都是聪明而勤奋的人，他们极富想象力，他们另辟蹊径，发现了计算机工程问题的巧妙解决方案。他们都有一个共同的价值观，那就是至今仍然长盛不衰的“黑客道德”。本书描述了近代历史上的一个萌芽时期，描述了黑客用默默无闻的行动为当今的数字世界照亮了一条道路，描述了那些打破陈规“非法”访问穿孔卡片计算机的MIT的学生，也描述了缔造出Altair和Apple II电脑这些伟大产品的DIY文化。

这是一本极好的书。它没有掺杂个人意志，作者只是站在旁观者的角度去描述那一段史诗般的岁月——而那些时光已经完全被现在的人所遗忘。阅读的过程就是不断震撼的过程，看那些纯粹的人不断地创造历史和挑战自我，仿佛可以呼吸到MIT科技广场大楼9楼的那种嘈杂的空气。然而这样的场景、这样的人已经很难再有了。为什么无法重现？

随着技术和产业的发展，以及计算机的普及（又是Hackers的功劳），也有人发现了其中蕴含的大量商机。这些人包括：内部的、外部的、恶意的。

最典型的内部例子是Symbolics公司，这个制造了最强大的Lisp机器的企业，不但从MIT分化成为纯商业公司，还不断地从9楼挖人。正好这些人很难拿到ARPA的资助了，于是开始进入商业的圈子。Gosper，这个领袖，就走了这样的道路。而现在，他一边做独立咨询的工作，一边继续玩数学、分形和Game of Life，从未磨灭的兴趣再也无法支撑自己的生活，让人唏嘘不已。

最典型的外部例子是Microsoft公司，完全不用多说。盖茨打破了Hackers最优秀的传统和约定，推动法律将代码保护起来并直接卖钱。这一产业开始形成，一方面大量的人发现赚钱是如此容易，另一方面自由交流带来的损失会如此巨大。

恶意获利的例子来得最晚。最早的恶意代码和恶意攻击都是圈子里的人在展现自己无所不能（就像在“9楼”做的事一样），甚至第一个rootkit就是Ken Thompson公布出来的。直到20世纪80年代末才出现ransom类的木马，加密了文件以后敲诈受害者，而大规模的获利性攻击直到90年代中期才出现。无论如何，这些恶意的获利者进一步促使了技术的封闭——虽然这种认为不被人知道就安全了的想法不怎么正确。

最早Hackers对硬件、系统、软件有着无与伦比的了解，主要原因是本质地简单。

不到五十年，一切都变了。再也找不到一个对硬件、操作系统、编译器/汇编器、软件、人工智能都熟练的人。这是任何一种知识发展的必然。

全球排行榜

Amazon

01 Cracking the Coding Interview: 150 Programming Questions and Solutions
02 Mastering VMware vSphere 5
03 Head First Java,2nd Edition
04 iOS Programming:The Big Nerd Ranch Guide
05 Artificial Intelligence:A Modern Approach
06 Objective-C Programming:The Big Nerd Ranch Guide
07 Programming in Objective-C
08 Head First HTML with CSS & XHTML
09 Head First Design Patterns
10 JavaScript:The Definitive Guide:Activate Your Web Pages

天珑书局（中国台湾）

01 前进Android Market！Google Android SDK实战演练
02 学徒模式：优秀软件开发者的养成之路
03 Eclipse完全攻略：从基础JAVA到PDE外挂开发
04 Google Android SDK开发范例大全
05 jQuery锦囊妙计
06 Google Android应用程序开发实战
07 Google！Android 3手机应用程序设计入门
08 ASP.NET 4.0专题实务－使用C#
09 一定要学会的HTML5＋CSS3网页设计实作应用
10 ASP.NET 4.0专题实务

第二书店

01 REST实战
02 C++反汇编与逆向分析技术揭秘
03 持续交付：发布可靠软件的系统方法
04 实用Common Lisp编程
05 深入理解Android：卷I
06 Java编程思想
07 Hadoop实战
08 高可用MySQL：构建健壮的数据中心
09 Debug Hacks：深入调试的技术和工具
10 松本行弘的程序世界

GEEK产品



I'm Watch

意大利的Blue Sky公司推出基于Android系统的I'm Watch系列手表，它可通过蓝牙装置和iOS系统以及Android设备实现同步。手表配备Freescale IMX233应用处理器芯片，1.54英寸的TFT触摸显示屏（分辨率为200dpi×200dpi），内置4GB存储空间和64MB的RAM内存。450mAh Li-Po电池可支持约3小时的免提通话或者约2天的待机。用户可以在手表上访问Facebook、Twitter等社交网站，收发E-mail、查看天气预报和浏览相册。





Nokia Luna蓝牙耳机

随着新手机的发布，Nokia同时也推出了Luna系列蓝牙耳机。这款产品具备NFC功能，通话时间可达8小时，即使是煲电话粥一族也可以勇敢尝试。耳机采用弹出式听筒设计，用户只需要轻轻一按耳机即可自动跳出，方便接听电话。耳机有五种颜色可供选择，在用完后插入底座即可充电，便捷又贴心。



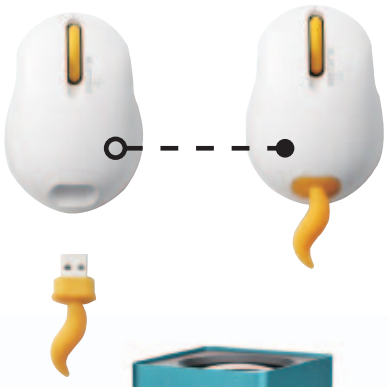
Finder

经常找不到钥匙？Finder来了！Finder是一个由主终端和定位贴两部分组成的RFID定位装置，包括主终端和标签贴。只需将定位贴贴在目标对象上，如钥匙、手机、钱包等，在主终端上一对一命名存储，要寻找时只需在主终端上输入名字，就能直观地显示跟踪物体所在位置。简单方便有效，再也不会丢三落四了。目前还是概念产品。



Oppopet无线鼠标

Oppopet无线鼠标将USB接收器做成了动物尾巴的形象。平时可将无线接收器插在鼠标屁股上，使用时拔下来插到电脑上，如此设计，无线接收器既不容易丢，还美观靓丽。每一款尾巴的造型都不同，有没有全收集齐了的冲动？共有狐狸、狗、猫、猪、兔子、海豚、松鼠、变色龙八款。



Lytro光场相机

Lytro是一家成立于美国硅谷的初创公司，凭借创始人在斯坦福上学期间研究的光场图像处理算法以及可工业化的光场相机原型，Lytro对现有的摄影技术产生了革命性的进展。光场相机通过内置的微型镜头阵列实现了一次拍照能够记录下系列景深的图像，进而实现一次拍照、多次对焦的效果。有人说第一代相机是胶片相机，第二代是数码相机，而第三代就是Lytro带来的光场相机。是否真的如此，让我们拭目以待。







Tagg Pet Tracker

还在为丢失宠物而担心吗？高通旗下公司Snaptracs推出了一个配备GPS和无线功能的高级宠物追踪系统——Tagg Pet Tracker，使宠物能与它们的主人保持联系。内置的锂电池可以连续30天发出信号，宠物主人只要在电脑或者智能手机上安装相关软件，就可以轻松找到宠物的具体位置，让主人安心。

行走在投资界的程序员

专访千淘资本合伙人李华兵

记者 / 陈秋歌

汽车、极客、足球、开放平台、风险投资、创业，这些是李华兵给自己的标签；汉能投资副总裁、千淘资本的合伙人、车品汇首席执行官是李华兵职业生涯的有力诠释。在本篇中，我们将探寻他从技术研发到创业的故事，讲述他跨越两个行业的程序人生。

在十多年的从业经历中，李华兵跨越了技术研发与投资两大行业。他结合在两大行业中的专业优势，打造了一条属于自己的职业发展之路。

上大学是儿时的梦想

李华兵出生于湖北鄂州的一个小山村。父亲作为当时村里唯一的工人，对他童年的成长有着很大的影响。父亲年轻时曾因学历低而影响了在部队的发展前途，不得不转业到地方工厂，对此父亲一直很懊悔。所以，父亲从小就教育李华兵好好学习，读书才是唯一的出路。

李华兵的家乡地处长江中下游丘陵地区，十分贫穷，很多同学在初中甚至小学就辍学务农、外出打工。家境虽然很艰苦，但在父亲的鼓励下，李华兵一直把学习放在第一位，考上理想的大学是他小时候的梦想。

“我成绩一直很不错，但我很叛逆，从来都是老师既爱又恨的学生”，李华兵这样评价儿时的自己。从小，他就个性鲜明，凭着不服输的劲头儿，他成功考入了武汉大学。李华兵对新鲜事物天生有着特别的好奇心，敢于尝试与挑战的精神，稀里糊涂地选报了计算机专业。



从技术研发到创业，从读MBA深造到投资，李华兵的职业经历值得学习和分享

社会实践是大学里的主要功课

大学时代应该是放下高考的压力与紧张、享受高校美好生活的时候，而李华兵并没有任何放松。他开始思索上大学的目的，并通过大学平台，获得人生的再一次飞跃。

武汉大学是一个人文气氛浓厚的院校，李华兵作为班长及学生会干部，积极参与组织学校的各种活动，他个人也经常参加一些论坛、讲座、大合唱等校园活动。相比这些校园活动，他更喜欢投入到丰富的社会实践中。利用暑期时间，他曾帮别人组装电脑，做过小商贩、配送员，发过传单等。这些看似枯燥、没太多技术含量的临时工作，却大大丰富了李华兵的社会阅历，练就了他坚韧的性格，为他日后步入社会，以独特全面的视角看待问题奠定了坚实的基础。

获得全职的实习机会，更好地了解企业的运营环境，一

直是他所渴望和追求的。1998年底，还在读大四的李华兵终于获得了这样的机会。在校园招聘时，他成功应聘为深圳市特发集团的开发人员，开始了企业实习。能去当时还很“神秘”的深圳工作，他异常兴奋，有一种展翅放飞、自由翱翔的激动。

两年内蜕变为技术管理者

毕业后，李华兵正式入职深圳市特发集团。在那里，他首次接触到商业程序，了解到软件开发的整个流程，在技术上获得了迅速成长。

深圳特发集团是一家国有企业，很多机制相对呆板。为了寻找更大的工作挑战和发展空间，工作近九个月后，他被一位武大师兄邀请到了深圳远望城多媒体电脑有限公司。这是一家民营企业，专注于软件开发和系统集成。如果说第一份工作练就了李华兵的技术功底，那这份工作则充分证实了他的技术实力。

在这里工作的一年半时间里，他自始至终参与开发了全国建设局第一套招投标系统，获得了客户的高度赞许。他由一名程序员，蜕变成项目管理者。从需求分析、文档撰写、代码编写到数据库设计一整套软件开发流程工作，他已能独立担当，开始肩负企业技术大梁，而这个成长过程仅仅花了不到两年的时间。

有理想之人总会追求更广阔的天地。

2001年，李华兵受另外一位武大师兄邀请来到北京，共同创办了北京光彩在线网络技术服务有限公司，这家公司是中国最早的一批B2B电子商务网站之一。“当时马云也才刚刚崭露头角”，李华兵如是说。作为技术总监，李华兵负责网站的整体搭建工作，而当时使用的技术体系就是.NET体系。“可以不夸张地讲，也许我是当时使用.NET技术体系来构建商务应用的最早一批人”，李华兵回忆说。然而该公司运营3年后，因为管理经验的缺乏和推广策略的失误，最后倒闭了。但作为首次创业，它给李华兵留下了刻骨铭心的印象，也为他后来的再次创业积累了宝贵经验。

创业失败后，他又被聘请到8848网站，负责研发该网站的竞价排名系统。

转身投资，迎来职业生涯的转轨

除了具有深厚的技术背景外，李华兵还是一名投资者。两种行业背景的结合，为他开创了一条与众不同、精彩纷呈的职业发展之路。

2004年，李华兵离开了8848网站，开始攻读清华大学国际MBA。他之所以选择MBA，一方面是为了充实自己，更重要的一方面是在工作期间，他发现风险投资是每个成功上市的互联网公司背后的坚实支柱，而当时绝大多数投资人都有MBA背景。

在李华兵看来，技术仅仅是一个公司成功的必要条件，但技术再牛也不能解决商业模式不对的问题，他希望转行做投资。

作为清华大学的交换生，李华兵于2005去了法国巴黎高等商学院（HEC）继续深造。在出国的这段时间里，李华兵周游了欧洲很多国家，对欧洲的风土人情有了清晰的认识，并形成了自己的外交方式。“这段出国经历，不仅使我的专业技能有了很大提升，还大大开阔了眼界，为我带来足够的自信，去迎接毕业后的第一份工作，”李华兵回忆说。

MBA毕业后，李华兵顺利入职汉能投资集团，迎来了职业生涯的顺利转轨。在汉能集团，他主要负责TMT行业的投资机会，期间操作成功投融资及并购交易累计金额达数亿美元，还曾帮助老牌中关村企业新奥特首次与国际资本碰撞，为它找到新的发展目标，也因此获得该企业领导的敬佩。

如何判断一个公司的价值？对于投资方来讲是很重要的。在负责投资银行业务的同时，李华兵对此也积累了一定的经验。他认为对于成熟公司和初创公司，判断标准是有区别的。对于成熟型企业，可以根据财务报表，对它所处行业的发展空间及企业未来发展趋势作出评价；而在早期企业，最重要的是对人的判断，再者是对创业行业的评估。

2009年，在毕业十周年之际，李华兵怀揣再次创业的梦想，离开了汉能集团，与人合伙创办了千淘资本，从事早期投资活动。车库咖啡、推图这些名字的背后都有李华兵的身影，而车品汇则是李华兵从零开始创建并亲自运营的一家汽车电子商务公司。

结语

从毕业后从事技术研发到第一次创业；从读MBA继续深造到转行投资，到再创业，可以看出李华兵丰富而曲折的职场经历。作为创业者，他认为创业是一个相当艰苦的过程：创业者必须具有很强的抗压能力，这是创业的核心；创业者要坚持正确的创业方向，顺势而为；创业者要有分享精神，采用合理的激励机制（如股权分配），以吸引更多的优秀人员加入；创业者要有很强的沟通能力，以此撬动更多资本资源，助推企业迅速发展。

Mac OS X 背后的故事（七）

Jean-Marie Hullot的Interface Builder神话

文 / 王越

Interface Builder，是用于苹果公司Mac OS X操作系统的软件开发程序，Xcode套件的一部分，于1988年创立。它的创造者Jean-Marie Hullot自称是“一个热爱旅行、充满激情的摄影师”，本篇分享Hullot热爱技术的那一面——创造Interface Builder的过程。

因势而动

1981年， Jean-Marie Hullot拿到巴黎第十一大学的计算机科学博士资格后，开始了法国国家信息与自动化研究所（INRIA）的研究生活。

20世纪70年代初，正是面向对象程序设计开始走上历史舞台的时期。许多现代计算机技术的诞生地Xerox PARC（施乐帕洛阿尔托研究中心）的Alan Kay、Dan Ingalls、Ted Kaehler 、Adele Goldberg等人，从1969年开始研发一款面向对象的程序语言Smalltalk，并于1980年正式公布。这是一个完整地实现面向对象范型的编程套件，包含了一种面向对象的程序设计语言、一种程序设计库和一个应用开发环境（ADE）。

虽然当时的机器跑得巨慢无比，但Smalltalk先进的思想对其他众多的程序设计语言（Objective-C、Actor、Java和 Ruby）的产生起到了极大的推动作用，对计算机工业界的发展产生了非常深远的影响。我们将会在今后介绍Objective-C时，详细介绍Smalltalk及其对Objective-C的影响，这里先一笔带过。

Smalltalk的发布在业界一石激起千层浪，也给Jean-Marie Hullot幼小的心灵带来了巨大的震撼。他立即明白了面向对象思想所代表的先进生产力，一定会改变今后数十年的程序设计流程，他毫不犹豫地成为面向对象编程模式的早期粉丝。

SOS的助力

那时，Jean-Marie Hullot使用早期的Macintosh计算机进行开发。不过他很快就和其他开发者一样，发现虽然Mac的用户界面做得不错，但开发程序实在是太糟糕了。他说：“当Macintosh被发明出来时，计算机和先前就大不一样了，你至少需要花60%~70%的时间在用户界面部分的代码上。”在Macintosh被发明之前，用户界面是相当简单的，只需要在命令行下面打一串字符，计算机就会回应出一行行的信息。所以在那个时代，开发者完全不需要专注于用户界面。而Mac一经发布，随之而来的众多的窗口和菜单，让整个世界都不一样了。虽然对于使用最终产品的用户而言是简单方便的，但对于码



Jean-Marie Hullot的名字似乎不为大众所熟知，但他设计的Interface Builder却深入人心，创造了一个个软件神话。

工来说简直是个噩梦。每次他们需要一个窗口或者菜单，都要从零开始构建。

聪明的Hullot开始动脑筋改进Mac编写用户程序难的现状。他开发了一个程序，有点像现在Windows系统中的“画板”。一侧的工具条，是类似菜单这样的大量可重用的对象；而另一侧，则是程序员想构建的用户程序界面。只要把工具条上的工具拖放到程序界面中，那么类似“打开”、“打印”等相

关的功能，就可以被添加到用户界面中。事实上，这个程序，是最早的一批能通过鼠标把控件拖入界面设计窗口实现相应功能的商业程序，是用户界面设计软件的先驱。

这个跨时代的发明被称作SOS，用Lisp语言编写【注：What are we going to called this thing中认为此时就是Interface Builder，但据The NeXTonian等多处资料表明，在Steve Jobs见到以前，该程序名为SOS】。当时，ExperTelligence开发了一种叫做ExperLisp的方言，SOS即用此语言写成【注：http://en.wikipedia.org/wiki/Interface_Builder】。

此时Hullot忽然意识到，他设计的东西事实上很强大，其重要性简直可以和Smalltalk这样的发明相比——Smalltalk让开发者尝到了面向对象语言的甜头，而SOS则是直接把对象放到了开发者手边。有了这么拽的东西，Hullot意识到如果他只在研究所窝着，那只能让十几个人享受这一成果，而如果他跳槽，把这个工具公开，那对天下的码工来说可是大福音。

诞生之源

经过不断努力，Hullot找到了一个值得推销自己发明的好地方——剑桥的苹果大学联盟（Apple University Consortium）。这个苹果和大学合作的组织看到Hullot的创作后反响很好，就推荐他去见Jean-Louis Gassee。 Jean-Louis Gassee是个法国人，时任苹果开发研究院主任，见到SOS后也认为这是个好东西，便说服他去美国闯一闯。经过几次的鼓励和推荐，加上美国对Hullot来说又不陌生，于是他就买了机票跳上飞机就奔赴美国。

不过当Jean-Marie Hullot来到美国加州苹果总部时，他却认为这不是一个工作的好地方——苹果已经是一个很庞大的企业，很难再有所创新发展。他最终决定不留在那儿，转而在美国寻找一个能把这个产品卖出去的人。四处推销之后，找到他用来写SOS的Lisp解释器的生产商，就是刚才提到的位于Santa Barbara的软件公司 ExperTelligence。

事实上，当时的ExperTelligence正在寻找合作商卖自己的Lisp，而Hullot也在找合作商卖自己的 SOS，两者一拍即合，随即打电话给 NeXT，共同推销自家的产品。

NeXT在Palo Alto总部的产品市场部人员接待了Jean-Marie Hullot和两位来自ExperTelligence的员工，被SOS的理念镇住，遂打电话请Steve Jobs下来看。Jean-Marie Hullot像复读机一样又把自己的大作秀了一遍。老谋深算的Steve Jobs事实上早就看中了SOS，但他对ExperTelligence的Lisp一点兴趣都没有。所以他装作对这场演示毫无兴致【注：这有很多引用该文的翻译译错，原文说nonplussed，字面意思为惊异，但在美国非正式表述中，此字表毫无兴致】，挥挥手就把这三个人打发走了。

但当他们一行人走到停车场时，Steve Jobs让他手下把Hullot追了回来，当他只身回到NeXT总部时，发现Steve Jobs正恭敬地等着他。

“我想要你计算机上那个程序”【注：http://rixstep.com/2/0/people/】，Steve Jobs说道：“你大概什么时候能开始给我们工作？”

Hullot回答说自己翌日就要离开去度假。

“好吧，我两周后给你打电话，”Steve Jobs说。

“不行，老乔”，Hullot表示：“我不游美国，我可要环游欧洲，你七个礼拜后再打给我吧。”

Steve Jobs虽然一骨子傲气，但他明白一个简单的道理：21世纪最缺的是什么——是人才！即使Jean-Marie Hullot玩起了大牌，这电话自然还是要打的。Hullot刚一度完假回来，Steve Jobs的电话就如期而至。

如此三顾茅庐般的热情，把Jean-Marie Hullot感动得第二天就登上了去美国的飞机。合约签了半年，但实际上他最终在NeXT整整待了十年。在NeXT工作期间，他使用Objective-C和NeXTSTEP框架重写了SOS，命名为Interface Builder。由此，Interface Builder成为NeXT集成开发环境 Project Builder标准套件之一。

进阶与探索

Interface Builder和SOS一样，提供了一个工具箱，包含一系列用户控件对象。工具箱并不是官方定死的，而是可以任意扩展的，比如如果用户想使用类似Safari中的toolbar，而这不是官方提供的，则下载第三方的PSMTabBar即可实现，甚至连Cappuccino这样的网页框架也可以用Interface Builder来完成设计。开发者只要把控件比如菜单和文本框拖入项目文件就能完成用户界面设计，节省了几乎所有和控件放置有关的代码。

开发者拖拽鼠标，将控件可提供的动作（IBAction）和另一个对象的接口（IBOutlet）连在一起， 则建立了一个绑定。这样，一旦动作被激发（比如用户点了按钮），那接口中相应的方法则会被执行。所以，大量对象关联的代码也能被省去。

有了这样的模式后，Interface Builder和Cocoa可以比后来出现的Microsoft Visual Studio或Qt Designer等软件走得更远——只要是对象，Interface Builder就能够操控它们，不需要一定是一个界面的控件。比如，数据库的数据源、队列等，都可以在Interface Builder中连接起来，于是很多

原本需要上千行的复杂应用（比如用来显示、修改企业中职工姓名、部门、电话、地址、头像等信息SQL数据库的用户界面程序），数分钟内就可以写完，不用一行代码。不信？让1992年的Steve Jobs亲自做给你看【注：http://www.youtube.com/watch?v=j02b8Fuz73A，第23分钟～第29分钟】。

NeXT被Apple收购后，苹果把下一代操作系统建立在NeXTSTEP的基础上。Objective-C和Cocoa被作为主要框架，而Interface Builder和Project Builder也因此受到重用。就官方的工具箱而言，支持Objective-C/Cocoa、Carbon的HIToolbox和WebObject。2008年3月27日，苹果发布首个iPhone SDK，设计Cocoa Touch界面的，也正是Interface Builder。可以说，Interface Builder一直随着公司产品的发展而不断拓新。

Jean-Marie Hullot是在NeXT被收购时进入苹果的。Steve Jobs令他率领在法国的一个小团队，秘密为Mac OS X 10.2开发一个办公软件。以往这样量级的程序，都是由苹果加州总部的大班人马完成。而这次，为了向世人表明他的Interface Builder有多强大，iCal横空出世，展示复杂的界面元素（日历、可拖拽的任务、五花八门的分类）和诸多功能（网络同步、Apple Script脚本控制）可以用相当快速的时间内开发出来【注：http://www.appleinsider.com/articles/07/10/17/road_to_mac_os_x_leopard_ical_3_0.html&page=2】。

最后，在iCal小组打完酱油的Jean-Marie Hullot荣升苹果软件开发部首席技术官。

Project Builder在Mac OS X 10.3时被重命名为现在大家所熟知的 Xcode。Xcode 3以前，Interface Builder使用一种名为nib格式的二进制文件格式。不过由于nib不能用肉眼读，也不方便使用版本管理工具来管理，所以Xcode 3开始新加入一种名为xib的文本文件格式，最后再在项目编译阶段输出为nib格式。和产生静态界面布局代码的工具（如MSVC、QtDesigner、 Delphi等类似的软件）很不同，nib是不被转译成相应Objective-C代码的。用户程序执行时，nib文件被读入，解包，并且唤醒【注：awake，即载入 nib 会自动调用程序中awakeFromNib方法】，所以nib文件是在运行时动态加载的。

长期以来，Xcode环境和Interface Builder是两个独立但相互工作的程序。而2010年释出的Xcode 4预览版中，Xcode和Interface Builder合二为一，成为一个一体化的编程环境。所以现在，开发者甚至可以只用鼠标在用户界面和代码间来回拖拽就能完成，这样一来Interface Builder对用户代码的解释也比先前更正确。比早期分离的程序使用起来确实方便很多。

当然，一个负面的影响是，这样用一体化集成开发环境写程序，往往会发现屏幕空间是不够的，所以像我这样用11寸Air或者13寸Macbook Pro的人，出去打招呼都不好意思说自己是做Mac开发的。

下一个海阔天空

在而后的岁月里，Interface Builder创造了一个又一个应用软件神话，小到官方教程中的汇率计算器，大到苹果所有的家用、专业软件，都由Interface Builder完成。

在风起云涌的1989年，欧洲核子研究组织（CERN）工作的科学家Emilio Pagiola忽悠经费，买来研究所的第一台NeXT计算机——当时NeXT计算机在CERN可是个新鲜事物——那里的科学家们纷纷前来把玩，普通青年发现里面有全本的韦氏词典，并可自动检查用户输入的拼写错误，技术青年发现它跑的是Unix系统，还有一个可读写的光驱，文艺青年更是发现里面居然预装了莎翁全集。不过毕竟像Emilio Pagiola这样忽悠巨款买NeXT机器的青年不多，所以大家围观完了，也就回去该干嘛干嘛了。

但Tim Berners-Lee和别人不一样，他不仅围观了那台计算机，还看到了Jean-Marie Hullot设计的 Interface Builder，研究了Objective-C，发现了面向对象编程范式开发环境的最高成就。这情景让他心中漾起了巨大的波澜，最终化为激情澎湃的投入，汇成了一行行面向对象的代码，一泻千里，奔向未来。

一年后，世界首个 HTTP 服务在CERN的NeXT计算机运行起来，而使用Objective-C和Interface Builder所编写的超文本语言编辑器兼浏览器同步发行。他给这个主从式架构起了个好听的名字——World Wide Web（万维网）。



王越

美国宾夕法尼亚大学计算机系研究生，中国著名TeX开发者，非著名OpenFOAM开发者。

责任编辑：白羽中（baiyz@csdn.net）

云时代的弄潮儿 Marc Benioff

文 / 刘明



云时代创业

在大多数人还不知道云计算为何物、互联网泡沫正面临崩溃的1999年，Marc Russell Benioff在旧金山租用了一间小小的旧公寓，创办了自己的云计算公司Salesforce.com，并把这个网站的使命定义为“End of Software”（终结软件）。而在短短的十几年内，它已成长为世界上发展最快、最具创新力的产业变革的驱动者之一。

Benioff是云计算的先驱，他除了一直致力于传播SaaS（Software as a Service，软件即服务）的理念之外，也是PaaS（Platform as a Service，平台即服务）一词的创造者。作为云计算布道者，Benioff出版了三本书来宣传自己的理念，这其中就包括畅销书《Behind The Cloud》（《云攻略》）。Salesforce与Amazon和Google被认为是云计算的三大领导企业。

生平经历

1964年，Benioff出生于美国加利福尼亚州一个富有创业基因的家庭。身为律师的祖父是湾区快速交通系统的发明者，而父亲则经营着自己的女装连锁店。他在中学时期就与同学一起创办了名为Liberty Software的公司，专门针对Atari系统生产和销售游戏，祖母负责制作音乐。1984年，他利用暑假在苹果公司Macintosh计算机部门实习，做汇编语言程序员，亲眼见证了乔布斯领导下苹果独特的创新文化。与此同时，Benioff还在南加州大学主修公司管理和科技创新课程。

大学毕业后，Benioff结束了自己的兼职创业生涯，进入甲骨文公司积累工作经验。知道吗，他的第一个工作是接听公司的800热线电话！但他很快爱上了这个职业，发现同客户打交道很有意思，而且与编程相比，自己更擅长与人打交道。出于对苹果的热爱，他总是不失时机地向Larry Ellison兜售自己开拓苹果市场的计划，并很快被后者提升为甲骨文Mac部门的主管。年仅23岁，他就被授予Oracle最佳新秀的荣誉。3年后，又晋升为副总裁之职，成为该公司最年轻的拥有这一头衔的人。在甲骨文的经历让Benioff深刻地理解了管理软件，并且对传统企业软件的定论“拥有才能控制”有了不同的想法。这成为他创办Salesforce的主要原因。

有意思的是，Larry Ellison对Benioff的创业非常支持。他除了投资200万美元外，还允许Benioff上午在Salesforce工作，下午来甲骨文上班，并在90天后新公司财务可以独立时建议他辞职，还承诺如果日后公司经营不善，还可以回甲骨文。他甚至同意Benioff挖三个甲骨文的员工。

2001年互联网泡沫破裂，许多网络公司用户流失，Salesforce遇到前所未有的严峻挑战。但塞翁失马，经济不景气使许多大公司转向全新模式的Salesforce。通过创新的技术模式、营销模式，Salesforce成为第一家年营收突破10亿美元的云计算公司。如今它每天要为十多万客户处理超过360亿宗交易。

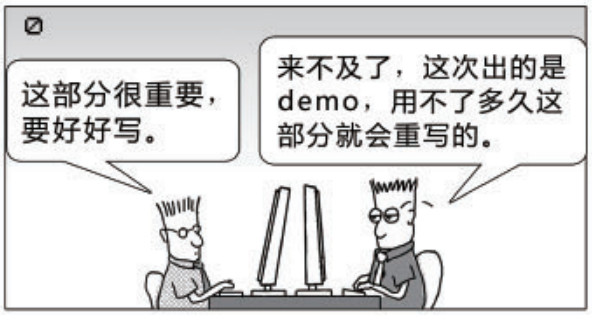
热心慈善

在云计算领域取得成功的同时，Benioff也不忘社会公益，开创了“1/1/1”的综合慈善模式——公司贡献1%的利润、1%的股权和1%的人时回馈给社区。这影响了很多公司，包括像Google这样的超级公司也采用了类似的模式。正因为他在慈善领域的贡献，世界经济论坛在2005年授予他“全球青年领袖”的称号。2010年，Benioff和妻子Lynne宣布捐赠给UCSF的儿童医院1亿美元，也因此被《Barron》周刊列入年度25位最有效率的慈善家。

云的未来

Salesforce的最新动作是收购Heroku，推出Face.com，在实现了Salesforce.com作为SaaS的同时，配合以PaaS。Benioff对社会化技术也非常重视，企业级微博Chatter和与丰田合作的车主社交网Toyota Friend都是这一思路的体现。有人评价Benioff正在打造整个互联网的后端，当然，这才是生于云时代的弄潮儿应有的作为。











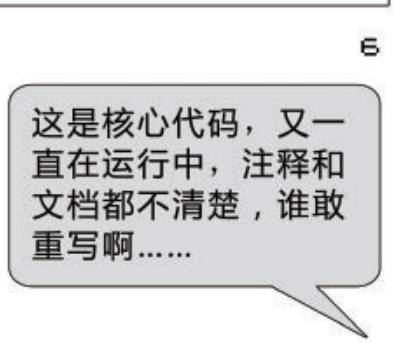





```
mm_segment_t old_fs = get_fs();
ssize_t result, copied = 0;

if (oldval && oldlen) {
loff_t pos = 0;
char buf[15], *nodep;
unsigned long area, node;
__le16 dnaddr;
//这部分必须在2010年5月前重构
set_fs(KERNEL_DS);
result = vfs_read(file, buf,
sizeof(buf) - 1, &pos);
set_fs(old_fs);
```







作者:
西乔

设计师，项目经理。06年起携创业团队从事Web技术外包开发及产品咨询顾问。

如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，请给西乔发邮件: arthur369@gmail.com

